$12.00

14th

ARRL DIGITAL COMMUNICATIONS CONFERENCE

Arlington, Texas
September 8-10, 1995

14th ARRL DIGITAL COMMUNICATIONS CONFERENCE

Co-Hosted by: Tucson Amateur Packet Radio (TAPR)
Texas Packet Radio Society

Printed in USA

ISBN: 0-87259-526-9

ARRL Order Number: 5269

First Edition

Foreword

The American Radio Relay League is pleased to offer these papers of the 14th ARRL Digital Communications Conference.

The papers presented this year focus on subjects as diverse as network enhancement, graphical information systems and much more. In 1995 some of the best minds in the Amateur Radio digital world have combined their knowledge to advance the state of the art--and you'll read the results in this book.

For example . . .

Discover how to squeeze seventy 9600-baud packet channels within the 2-meter band. And if you have the need for speed, check out the description of WA4DSY's *56 kilobaud* RF modem.

If you're putting together a packet network, you'll get valuable tips here--including information on DAMA, an innovative solution for crowded frequencies. You'll learn how TexNet is interfaced to the National Weather Service in Tulsa, Oklahoma, as well as an update on TexNet in general. If your plans include TCP/IP, read about how they're using it in the Puget Sound Amateur Radio network.

Did you know that you can model data communications functions using Microsoft *Excel*? Read N5EG's paper and see for yourself. While you're at it, read the fascinating discussion by Phil Karn, KA9Q, of convolutional decoders for packet radio.

Need I say more?

If you can't attend the conference yourself, this is the next best thing. These discussions should whet your appetite for experiments of your own. All you need is an ample imagination and the will to explore new territory.

As in previous conferences, all papers in these proceedings are unedited and solely the work of the authors.

David Sumner, K1ZZ
Executive Vice President

Newington, Connecticut
August 1995

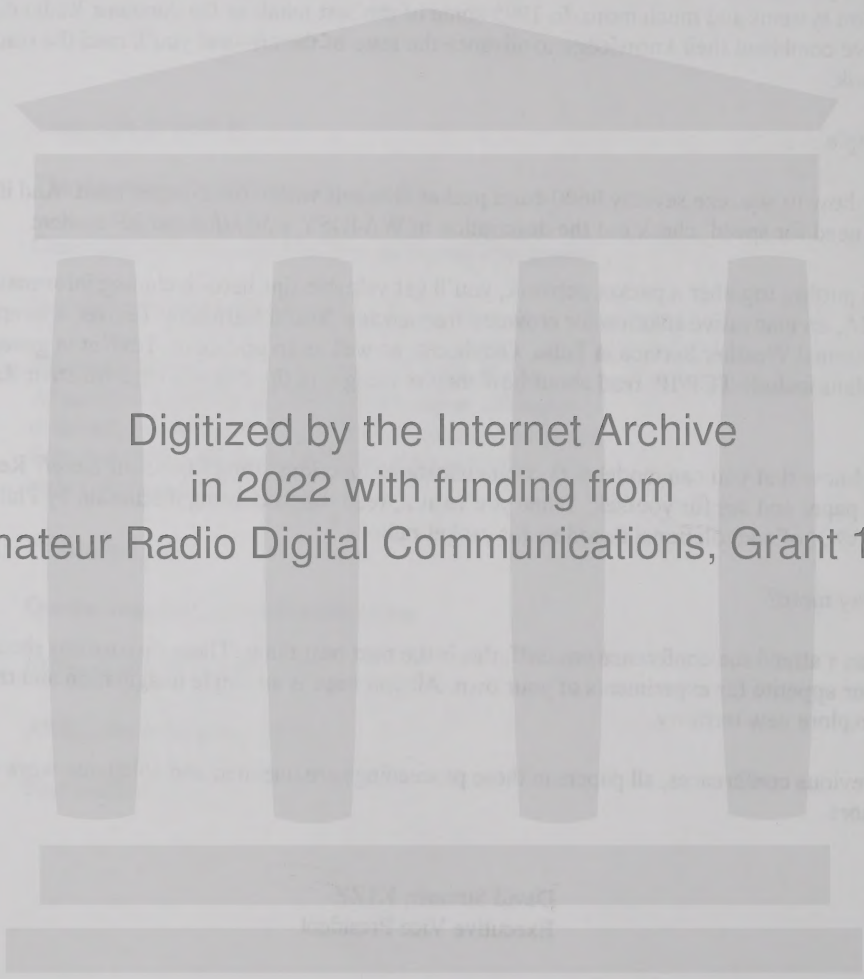


Table of Contents

Availability of Seventy 9600 Baud Packet Channels on Two Meters
Bob Bruninga, WB4APR 1

The WA4DSY 56 Kilobaud RF Modem
Dale A. Heatherington, WA4DSY 4

Extended Sequency Number (Modulo-128) Option for AX.25
Rob Janssen, PE1CHL 16

DSP-93 Update: The TAPR/AMSAT Joint DSP Project
Greg Jones, WD5IVD, Bob Stricklin, N5BRG, 20
and Robert Diersing, N5AHD 20

Introduction to Programming the TAPR/AMSAT DSP-93
Ron Parsons, W5RKN, Don Haselwood, K4JPJ,
and Bob Stricklin, N5BRG 27

An Introduction to FlexNet
Gunter Jost, DK7WJ 37

Convolutional Decoders for Amateur Packet Radio
Phil Karn, KA9Q 45

Data Radio Standard Test Methods
Burton Lang, VE2BMQ, and Donald Rotolo, N2IRZ 51

Modeling Some Data Communications functions Using Microsoft Excel 5.0
Thomas C. McDermott, N5EG 56

Building a Packet Network
Karl Medcalf, WK5M 65

DAMA—Another Network Solution
Karl Medcalf, WK5M 73

The Tulsa National Weather Service TexNet Interface Project
Bob Morgan, WB5AOH, and Greg Jones, WD5IVD 77

An Update on TexNet and the Texas Packet Radio Society
Bob Morgan, WB5AOH, and Greg Jones, WD5IVD 86

DSP-93 Programming Hints
Frank H. Perkins, Jr., WB5IPM 97

NETMGR: A Graphical Configurator for ROSE X.25 Packet Switch Networks
William Slack, NX2P, and Donald Rotolo, N2IRZ 101

Graphical Information Systems and Ham Radio
Keith Sproul, WU2Z 108

AX.25 Transport Layer Drivers for TCP/IP
Mark Sproul, KB2ICI, and Tim Hayes, N2KBG 119

The Puget Sound Amateur Radio TCP/IP Network
Steve Stroh, N8GNJ 125

6PACK—A "Real Time" PC to TNC Protocol
Matthias Welwarsky, DG2FEF 132

Skimming the Layers
Ken Wickwire, KB1J 137

Proposed Recommendation for Hierarchical Addressing Protocol
Dave Wolf, WO5H, Greg Jones, WD5IVD,
Roy Engehausen, AA4RE, and Hank Oredson, W0RLI 166

AVAILABILITY OF SEVENTY 9600 BAUD PACKET CHANNELS ON TWO METERS

By Bob Bruninga WB4APR @ WB3V.MD

Unbelievable? Not really! With the advent of the latest 9600 baud packet radio modems, there is an un-exploited mechanism for opening up dozens of half duplex data channels without ANY impact on existing voice and data bandplans. Read this proposal thoroughly before jumping to any conclusions. First you must consider two apparently un-related facts:

TINY PACKETS: With the new 9600 baud packet modems, not only is there potential for higher speeds, but the packets are also 8 times shorter than conventional packets. This means that each packet occupies the channel for less than 1/10 of a second at a time. If another signal appears on the data channel, then the presence of the new signal can be detected in 0.1 second.

AVAILABLE CHANNELS: Throughout the two meter band, in every corner of the country, there are almost 70 FM channels assigned to single VOICE receivers that, in general, only use their single frequency about 1% to 40% of the time. If a mechanism could be designed to permit these single receivers to continue to use their frequencies on a primary basis, at any time, with priority access and control, then the rest of the time (60% to 99%) they could use their channel for moving digitial data. This mechanism could more than double the bandwidth available in the two meter band! (I used 2 meters only to get your attention. One hundred twenty channels on the 440 band is the actual target because of the availability of 2 Watt 9600 baud UHF data radios for under $150 each.)

VOICE/DATA CHANNEL SHARING: In the past, attempts to share voice and packet have all failed, not because it is a bad idea, but because it has not been done properly. Under a very unique set of conditions, however, voice and data can easily share a narrowband FM channel, IF:

1) VOICE has priority at ALL TIMES and all voice operations are completely transparent to the voice user
2) DATA can never interrupt or attempt to use a busy channel
3) VOICE users can pre-empt/interrupt data instantly at ANY time
4) VOICE users do NOT hear packets or in any way are encumbered by shared use.
5) NO MODIFICATIONS TO ANY voice radio is required

In other words, voice users do not even know that the channel is shared with data. To make data use of a channel work on a secondary basis (behind the scenes) with NO impact on voice usage, there are additional requirements:

1) There must be ONLY one voice RECEIVER on the channel, and it MUST be able to hear EVERY voice user on the channel.
2) This single voice receiver and its conventional COR or un-modified squelch circuit must have total control over channel use.

THE BEST KEPT SECRET: Think for a moment about the input frequency of a typical voice repeater. There is only one receiver listening, and it can hear EVERYONE that desires to use the repeater. If the repeater RECEIVER does not hear anyone using the channel, then IT alone can decide to use the channel itself for data! Now if the data is transmitted in 0.1 second bursts, FROM THIS SITE (on the input channel), with a pause in-between to listen for voice users, then no one will be denied access to the voice repeater for any longer than 0.1 second! Also, while the repeater is transmitting data on its input frequency, nothing is being transmitted to anoy users on the output! In this manner, we can TRANSMIT data FROM this repeater site at 9600 baud on the input frequency WHENEVER THE REPEATER IS NOT OTHERWISE BEING USED FOR VOICE.

MAKING A DATA LINK: Now combine one such voice repeater with another operating in the same manner, and you have a two-way data link between these two sites. Not only is this a FULL DUPLEX channel, but it also operates with NO HIDDEN transmitters, and NO CONTENTION, because there is only ONE transmitter on each such channel. From a data perspective, each repeater site is a data node that transmits on ONE assigned frequency and can LISTEN on as many additional channels as desired. This is an ideal multi-node backbone network for passing traffic point-to-point over long distances!

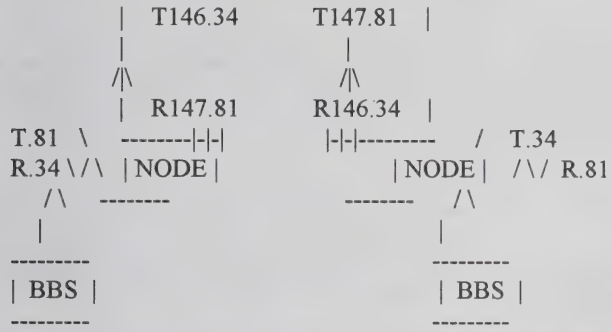
HARDWARE: To avoid even momentary delays during normal repeater conversations, the external carrier detect of the modem is not only driven by the COR of the repeater receiver, but also by the HANG-TIME of the repeater transmitter as an indication that the repeater is engaged in a conversation. This way, data will only be transferred after the repeater has been unused by voice users for a while.

FORWARDING: Obviously, this type of channel will NOT support any traffic where an impatient human is on one end. This channel, however, is ideal for off-hour BULK forwarding between multi-channel level-4 nodes under computer control.

BAUDRATE: Although I suggested off the shelf 9600 baud radios, the use of 15 KHz repeater channels in most areas of the country might require slowing to 4800 baud to be certain that all energy is contained within 12 KHz. Also, the $139 TEKK 2 watt UHF data radio was actually designed commercially for 4800 baud, and performs excsllently at that rate. Also, to avoid co- channel interference, operating at power levels above 2 watts is probably not advisable. Fortunately, 9600 baud data sounds just like white noise, and usually will NOT open the squelch of another repeater anyway. Finally, to experiment with this concept, you can actually try conventional 1200 baud during off hours. You can limit packets to a single frame (about .8 second instead of the usual 3 to 5 seconds), by setting MAXFRAME to 1 instead of the default 7. By adding an additional hang-time timer, so that the DATA mode is only activated when the repeater is unused for over 3 minutes, a new voice in-the-night will probably not even notice a maximum of 0.8 second delay.

SYSTEM LINKS: Remember, ONLY THE REPEATER ITSELF can transmit on its own INPUT frequency. The way to build a network is then to have each such repeater node TRANSMIT on its own input frequency and similarly LISTEN to the other nodes transmitting on THEIR input frequencies. To get data from your basement BBS, simply have it LISTEN to the repeater node on the input frequency and the BBS then TRANSMITS on its OWN unique frequency. The repeater node then listens on this unique frequency. If you are willing to cut throughput in half in order to save on hardware costs, you can let your BBS transmit (at low power) on the same frequency as some distant repeater node (its

voice input freq). This causes a hidden transmitter problem at your repeater which will hear both your BBS and the remote repeater, but with the tremendous bandwidth available, this is probably not a problem. The following diagram shows a typical arrangement for the nodes located at a pair of 146.34/94 and 147.81/21 voice repeaters.



Notice that by pairing up a 146 MHz repeater with a 147 MHz repeater, you get at least 1.6 MHz spacing between the two digital frequencies and about 1 MHz between the digitial and the Voice repeater output (which of course is NEVER transmitting when the digital is in use.

Also notice, that to save dual receivers at the NODE sites, we are cheating a PURE network design by allowing the BBS's to also transmit on the same frequecy as the DISTANT repeater node. This must be done with a beam or low power so that the distant repeater CANNOT hear nor even detect the presence of the BBS. Otherwise, the BBS would KEY UP the distant VOICE repeater!

CONCLUSION: Even if only 30% of the voice repeaters begin to share their channels, this could open up over 600 Kbytes PER SECOND of additional digital forwarding capacity on 2m and 70 cm! Why not?

The WA4DSY 56 KILOBAUD RF MODEM
A Major Redesign

By Dale A. Heatherington, WA4DSY

Abstract

In 1987 I designed a 56 kilobaud RF modem which was sold in kit form by GRAPES, the Georgia Radio Amateur Packet Enthusiast Society. This paper describes how the WA4DSY 56 kilobaud RF modem was radically redesigned to lower cost, reduce size, and improve reliability, manufacturability and useability. The reader is refered to the ARRL publication Proceedings of the 6th Computer Networking Conference, page 68 for details on the original design.

Overview

The original modem was implemented on 3 PC boards. It required both plus and minus 5 volts for the modem and 12 volts for the external transverter. The purchaser of the kit had to fabricate his own enclosure and obtain a suitable power supply. There were no status indicators. Only those hams with above average home brewing skills would attempt to build the unit. However, once built, the modem performance and reliability were quite good. Several high speed networks have been successfully built using these modems.

Unfortunately the large amount of skilled labor required to build a modem kit and to some extent, the cost have limited the wide spread adoption of these modems for high speed networking. I have redesigned the modem to address these issues.

The new design implements the modem on a single 4 layer printed circuit board powered from a 12 volt power supply. The PC board measures about 7 inches on each side. Signals produced by the new design are identical with the original and the new modems will interoperate with the old ones.

Most of the modem functions are implemented digitally in a Xilinx (tm) Field Programmable Gate Array (FPGA). The bandwidth limited MSK signal is generated digitally at 448 kHz to eliminate the need for analog double balanced modulators and a 90 degree phase shifter. This signal is up converted to the 10 meter band at 1 milliwatt. The converter is synthesized over a 2 mhz range (28-30 mhz). An external transverter converts the signal to the 222 or 430 mhz band. The delay between RTS and signal out is quite low, about 20 microseconds or 1 baud interval.

Note that this is a true *modem* which converts data to RF, unlike the G3RUH and K9NG designs which are baseband signal processors and don't do any *modulating* or *demodultating*.

The receiver is implemented with a single chip device and is synthesizer tuned from 28 to 30 MHz. A quadrature detector is used for FM demodulation. The demodulated signal is sliced using a circuit similar to the one in the original design which automatically adjusts the slicing level. The signal is then fed into the FPGA where clock recovery and data carrier detection are done digitally. The delay between receiving a signal and carrier detect indication is about 3 milliseconds.

The user interface has been greatly improved. Ten LEDs indicate received signal level. Other LEDs indicate *Request To Send*, *Data Carrier Detect* and *Ready*. The data interface is dual mode. A single switch selects CMOS or RS422 modes. The signals are presented on a DB15 connector wired to mate with the Ottawa PI2 Packet Interface Card. Other devices such as TNCs can be connected by wiring an appropriate cable and connectors.

Unlike the original design which allowed the user to reconfigure the modem for different baud rates, the data rate of the new design is fixed. Major changes are required to both the RF and FPGA circuits for use at any other baud rate. Part of the reason is because the first IF of 448 KHz must be 8 (or some power of 2) times the baud rate. Also, the receiver chip is being operated close to its maximum rate at 56KB.

Data Coding

All data coding is done in the FPGA chip. The chip is clocked at 14.31818 MHz. The transmit clock is obtained by dividing by 256. The exact baud rate is actually 55.9304 kilobaud, the same as the original design. The transmit clock signal is sent to the user on the RS422 interface. The user's transmit data source is expected to use the rising edge to clock out each bit. The modem samples data bits on the falling edge of the clock.

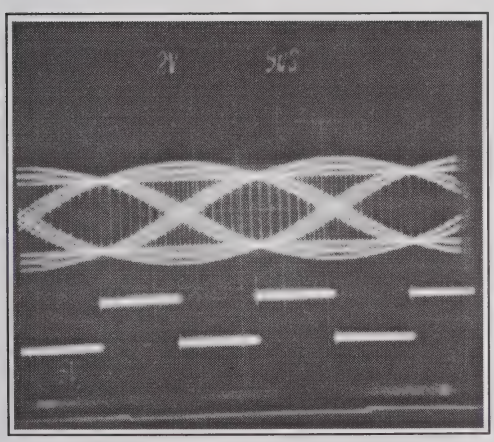
The transmit data stream is scrambled using the same shift register configuration as the original modem, a 17 bit register with feedback taps at stages 5 and 17. This is *not* compatible with K9NG and G3RUH 9600 bps modems. For more details on the scrambling system, see the section titled "Descrambler" below. After scrambling, the data enters the digital state machine where both NRZ to NRZI conversion and RF waveform table lookup operations are performed.

Modulator

This modem has no physical modulator. All RF waveforms are stored in EPROM. A digital state machine fetches the appropriate waveform segment from EPROM in response to the current data bit to be transmitted. 32 samples of the stored digitized waveform segment are read from the EPROM and sent to a Digital to Analog Converter (DAC) during each baud interval. The carrier frequency is exactly 8 times the baud rate to permit the splicing of different waveform segments together without phase discontinuties. The result is a 448 kHz bandwidth limited MSK signal. Unlike the previous version of this modem, there are no modulator related adjustment controls. The signal always has perfect phase shift and deviation characteristics. The signal is identical to the one produced by the original WA4DSY 56 KB modem.

Transmitted signal characteristics

- Modulation is MSK
- Bandwidth is 70 kHz at 26 DB down
- 3.5 DB amplitude variation
- 14 kHz FM Deviation
- 90 degree phase shift per baud



Top trace: Raw TX signal from 8 bit DAC

Lower Trace: 56KB TX Clock

Upconverter and IF

The 448 kHz MSK signal shown in the photo above is first filtered with an 80 kHz wide 3 section LC bandpass filter to remove digital sampling noise. It's then mixed with 10.245 MHz and converted to 10.693 MHz. The 10.693 MHz signal is passed through two 10.7 MHz (180 kHz BW) ceramic filters to remove the local oscillator and unwanted lower sideband (10.245 and 9.797 MHz). The undesired frequencies are reduced by at least 90 db.

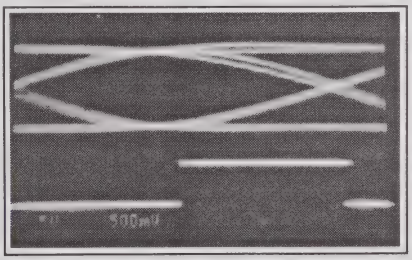
The desired 10.693 MHz signal is then mixed with a VCO signal in the 39 MHz range. The lower sideband of the mixer output (28-30 MHz) is selected with a two section LC bandpass filter. Both conversions are done with NE602 frequency converter chips.

The 29 MHz signal is amplified 30 db by an MMIC chip and sent out to the users transverter on a BNC connector. The output level is adjustable from -10 DBM to +5 DBM.

The local oscillators are running at all times to assure instant response to the user's "Request to Send" control signal. The total delay is less than 20 microseconds from RTS to RF data signal out. Contrast this to the original design which required up to 6 milliseconds to start the transmitter.

Receiver

The receiver uses a Motorola MC13135 chip for all RF signal processing. The received signal in the 28-30 MHz range from the user supplied transverter is filtered by a two section bandpass filter before entering the receiver chip. The first local oscillator is a VCO in the 39-41 MHz range controlled by a synthesizer. The signal is mixed with the VCO to convert it down to 10.693 MHz. The signal is bandpass filtered with a single 180 kHz wide ceramic filter before being mixed with 10.245 MHz and converted down to 448 kHz. A 60 kHz wide LC bandpass filter provides both selectivity and deemphasis. Frequency modulation is recovered with a quadrature detector.



Receiver Eye pattern

Frequency Synthesizer

I used a Motorola MC145162 synthesizer chip for this design. It is programmed serially with a three wire interface. It has completely separate reference counters and divide by N counters for transmit and receive. The

reference oscillator running at 10.245 MHz also drives both the receive down-converter and the transmit up-converter.

The VCOs are designed to cover 39 to 41 MHz, 10.693 MHz above the 29 MHz IF frequencies. The receive VCO is included in the MC13135 receiver chip. The transmit VCO is a Colpits transistor oscillator with an emitter follower output buffer.

Since there is no microprocessor in this modem, I needed a way to generate the data to program the synthesizer. I had the good fortune of having extra space in the EPROM; so I put the frequency data there. Up to 8 different bit patterns for independent TX and RX frequencies can be stored in the EPROM. The first thing the FPGA does after loading its configuration data is read one of the selected frequency bit strings into the synthesizer chip. Three switches code for the 8 frequencies. Any 8 frequencies can be programmed into the EPROM using a simple program written in C. The program will be supplied with the modem. If the user wants a custom set of frequencies, he must have access to an EPROM programmer or order a custom EPROM from the dealer. Still, this is superior to the original design which used custom crystals often requiring a 6 to 8 week wait and costing 10 to 15 dollars each.

Gated Tracking Data Slicer

Before the recovered signal can be used it must be processed to determine the state of the received bit, 1 or 0. This is done with an analog comparator chip. Its threshold is set exactly halfway between the voltage level of a "1" and a "0". It outputs a "1" if the input is higher than the threshold and a "0" if it's lower. There is a problem when the carrier frequency of the incoming signal changes. The voltage levels of the ones and zeros change so the threshold is no longer exactly half way between them. This causes an increase in errors. One common solution, which doesn't work very well, is to AC couple the output of the demodulator to the detector. This is fine if the short and long term average of the number of ones and zeros is equal. This ideal condition cannot be guaranteed even if a scrambler is used. A much better solution is to put some intelligence in the detector so that it averages the voltage level of the ones separately from the average of the zeros and then subtracts the two averages to obtain the ideal threshold level. This circuit doesn't care about the ratio of ones to zeros as long as there is a reasonable number of each. A scrambler is used to make sure there is a reasonable number of both ones and zeros. The circuit will compute the correct threshold if the input signal carrier frequency is anywhere within the expected range of the ones and zeros, in this case plus or minus 14 kHz. The data slicer used in this implementation is gated with the recovered clock. It only sees voltage levels near the center of a baud interval. A leaky sample and hold technique is used to grab the middle 1 microsecond of each bit. There is little variation in the peak levels from bit to bit thus reducing unwanted fluctuations in the slicing level.

Clock Recovery

Clock recovery is done digitally in the FPGA. There are no adjustments such as VCO center frequency as in the original design. The phase of a 3 modulus counter is compared with the data zero crossing times. The counter is driven by a 3.579545 MHz clock. The baud rate for each modulus is listed below.

Modulus	Baud Rate
• 63	56.818 (fast)
• 64	55.930 (on time)
• 65	55.069 (slow)

The counter can divide by 63, 64 or 65. The divide by 64 setting produces a 56 kHz clock. If the zero crossing was late relative to the counters terminal count then the counter is counting too fast. The counter modulus is set higher so it will be earlier next time. If the zero crossing is early the modulus is set lower so it will be later next time. If no zero crossing is detected the modulus is set to "on time" so the clock won't drift during strings of ones or zeros. This scheme only introduces about 0.5 microseconds of clock jitter (3%).

Data Carrier Detector

The data carrier detector is also implemented digitally in the FPGA. There are no adjustments. Two gates are used to separate data zero crossings which fall within plus or minus 12 1/2% (in sync) of the terminal count of the 3 modulus counter described above from the zero crossings which fall outside this range (out of sync). If the clock recovery circuit is phase locked, all zero crossings should fall within the 25% "in sync" window. This is true even at low signal to noise ratios. The "in sync" zero crossings cause a 5 bit counter to increment. The "out of sync" zero crossings cause the counter to decrement. The "carrier detected" flip flop is set when the counter reaches maximum count (31). The flip flop is reset when the counter reaches minimum count (0). The counter is designed not to overflow. It has "stops" at count 0 and 31. Carrier detect occurs when the clock recovery circuit has acquired phase lock and 31 more "in sync" zero crossings have occurred relative to "out of sync" zero crossings. This takes about 3 milliseconds or about half the time of the original 56KB modem with less falsing. Measurements show solid carrier detect even when the bit error rate is as high as 6%. Random noise can't assert carrier detect because the zero crossings have random timings and will occur with equal probably at any point in the baud interval. Since 75% of this interval is devoted to decrementing the 5 bit counter, it will quickly go to zero and reset the carrier detect flip flop. Periodic waveforms that are harmonically related to the 56 kHz clock frequency will trigger carrier detect if the clock recovery circuit phase locks to it.

NRZI to NRZ conversion

NRZ is a data signaling format in which zeros are represented by a certain voltage level and ones by another. NRZI is a signaling format in which zeros are represented by a change in voltage level while ones are

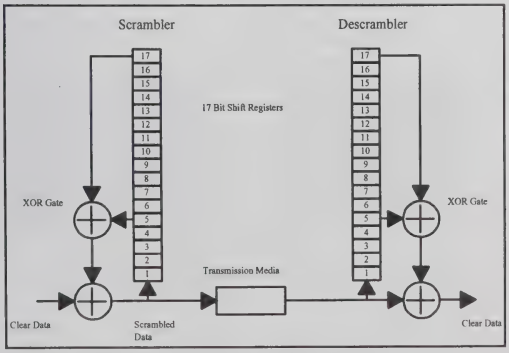
indicated by no change. NRZI coded data is not affected by inverting the data voltage levels or the mark/space frequencies in the case of FSK. This modem converts the incoming NRZI data to NRZ data with a simple circuit consisting of a "D" Flip Flop and XOR gate. These components are in the FPGA chip.

Descrambler

A self synchronizing data scrambler was used in this modem for two reasons. First, it makes the data stream look like a random stream of ones and zeros regardless of the data being transmitted. This characteristic makes the tracking data slicer and clock recovery circuits work better. Second, it makes the RF spectrum look and sound like band limited white noise. In other words, the RF energy is spread evenly over the modems bandwidth and shows no single frequency lines regardless of the data being transmitted. Any potential interference to neighboring channels is limited to an increase in the noise floor instead of squeaks, squawks, and other obnoxious noises. This type of scrambling is also commonly used in high speed synchronous modems for telephone use.

The hardware to implement the scrambler and descrambler is very simple. It consists of a 17 bit shift register and two XOR gates, also known as a *Linear Feedback Shift Register* (LFSR). Each transmitted bit is the result of the exclusive ORing of the current data bit with the bits transmitted 5 and 17 bits times before. To descramble the data, it is only necessary to exclusive OR the current received bit with the previous 5th and 17th bits. If the data consist of all ones, the scrambler will produce a pseudorandom sequence of bits that will repeat after 131,071 clock pulses or every 2.34 seconds at 56 kilobaud.

This linear feedback shift register scrambling scheme does not violate the FCC prohibition against codes and ciphers because its purpose is to "facilitate communication" and the algorithm is publicly available.



Scrambler Block Diagram

Note: G3RUH and K9NG scramblers use 17 bit shift registers tapped at stages 17 and 12. The sequence produced is not maximum length.

8 Bit FIFO and Bit Repeater Mode

To allow this modem to be used as a full duplex bit repeater I have included a first in-first out (FIFO) buffer and logic circuitry to route the received data bits back to the transmitter. When repeater mode is enabled, the data carrier detect signal will assert request to send. A 2 second watch dog timer prevents transmitter lockup. The FIFO buffer is 8 bits long. To allow for both plus and minus "bit slip", the FIFO does not start sending data until it's half full. The transmitter then pulls bits out of the FIFO at it's fixed clock rate while the receiver inserts bits into the FIFO at the receiver clock rate. The FIFO is reset when data carrier detect (DCD) drops. With a data rate difference between incoming and outgoing data of 0.01% a packet of 40,000 bits (5000 bytes) can be retransmitted before the FIFO buffer overflows. Since this is much larger than typical AX25 packets, I don't think this restriction will be a problem. Keep in mind that DCD must drop between packets to reset the FIFO.

A single internal switch enables the bit repeater mode. The external data I/O connector remains active to allow communication with a local computer.

Watchdog Timer

A 2 second watchdog timer is built in and may be bypassed with a switch setting. The timer is reset when RTS is false and begins timing when RTS is true. PTT will be turned off if RTS is not removed after about 2 seconds. A resistor and capacitor set the time-out value. This is the only analog timing circuit in the modem.

Remote Control

Computers used in amateur packet networks are often located with the TNCs, modems and radios in inaccessible places such as mountain tops. When the computer software crashes, which it often does, the control operator doesn't want to have to go to the distant site to push the reset button. A remote control reset function is built into this modem.

Normally open relay contacts are available for whatever use may come to mind. The contacts close when the modem decodes several milliseconds of a pseudorandom bit pattern sent to it from another modem in convenient reach of the control operator. A push button on the rear panel causes the modem to send the special sequence.

The sequence generator is an 8 bit linear feedback shift register with user specified taps. The tap locations are specified along with the frequency data in the EPROM. Each RX/TX frequency pair may have a different code assigned. Each set of taps produces a unique code. The receiving modem must see at least 256 bits of

the pattern before it will start to close the relay. The sequence must continue for another several milliseconds to allow the relay to close. A single incorrect bit in the sequence will reset the decoder so that another correct sequence of 256 bits are needed to cause relay closure. The front panel "ready" LED will change from green to red when the code is being received.

Signal Level Display

Ten LEDs on the front panel indicate relative signal strength. The RSSI signal from the MC13135 receiver chip drives an LM3914 linear bar graph display controller chip. I have found this a most welcome feature. I've used it to map signal coverage areas by first setting my station to "ping" the local 56KB packet switch every 2 seconds; then with only a modem, transverter and antenna in the car, I can get a good idea of how well the packet switch covers various areas by watching the signal level LEDs and the DCD LED (and the road).

Tune-up and Test Aids

Since only half the EPROM storage was used for the main FPGA configuration code, state and waveform tables, I provided a switch which allows the modem to "boot up" using the other half of the EPROM. In the other half is an FPGA configuration for a direct digital frequency synthesizer which is used to sweep the modems tuned circuits. It sweeps a 200 KHz range centered on 448 KHz. A square wave is also generated on the receive clock output line for scope sync. The rising edge coincides with the 448 KHz center frequency. Adjustment of the filters for proper response shape requires only a 30 MHz dual trace scope. The scope must be adjusted so the rising edge of the "receive clock" square wave is centered and exactly one complete cycle is displayed. The other channel can then be used to probe various points in the modem to observe the frequency response envelope calibrated to 20 KHz per horizontal division. The receive filter can be checked if the an attenuator is placed between the TX and RX BNC connectors. The transmitter becomes the sweep generator.

A push button on the rear panel will activate the transmitter and send scrambled marks. The 2 second watchdog timer is automatically bypassed to allow transverter tune-up or power measurement.

Interfaces

Transverter: Power and PTT (Push To Talk) transverter signals are provided on a 5 pin DIN connector. The remote control relay contacts are also on this connector. BNC connectors are provided for the 29 MHz transverter IF signals. Pin assignments are as follows:

1. PTT
2. Relay contact
3 Ground
4. Relay contact
5. +12 volts @ 2 Amps

Power: The power input connector is a common 2.1 mm round DC power jack used in many other consumer electronic devices. Positive voltage is supplied on the center pin. A 12 volt 2.5 amp external switching power supply runs the modem and transverter.

Data: A female DB15 connector is used for the data interface. The pin assignments are the same as the Ottawa PI2 card. A single switch on the PC board changes the electrical standard from balanced RS422 to unbalanced CMOS. The RS422 interface is based on the 26LS32 and 26LS31 chips. The CMOS interface uses a 74HC244 chip.

Pin assignments

RS422

1.	No connection	
2.	+ Receive Clock	(out)
3.	+ Receive Data	(out)
4.	+ Transmit Clock	(out)
5.	+ Carrier detect	(out, low true)
6.	+ Transmit Data	(in)
7.	+ Request to Send	(in, low true)
8.	Mode Select	(Not Used)
9.	Ground	
10.	- Receive Clock	(out)
11.	- Receive Data	(out)
12.	- Transmit Clock	(out)
13.	- Carrier Detect	(out, high true)
14.	- Transmit Data	(in)
15.	- Request to Send	(in, high true)

CMOS

1.	No connection	
2.	Receive Clock	(out)
3.	Receive Data	(out)
4.	Transmit Clock	(out)
5.	Carrier detect	(out, low true)
6.	Transmit Data	(in)
7.	Request to Send	(in, low true)
8.	Mode Select	(Not Used)
9.	Ground	

10..15 No connection

Internal Option Switch Functions

Switch	Off	On
1.	RS422	CMOS
2.	Normal	RX Mute Disable
3.	Normal	Repeater Enable
4.	Normal	Scrambler Disable
5.	Normal	Key Transmitter
6.	Normal	Tune up
7.	Normal	Watchdog Disable
8.	Frequency Select 2	
9.	Frequency Select 1	
10.	Frequency Select 0	

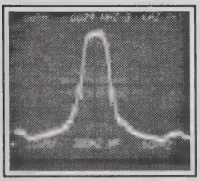
Performance

Due to deadline and and other time constraints, I was unable to do a bit error rate test on the latest PC board revision. The previous version needed about 2 db more signal to achieve the same bit error rate as the original WA4DSY 56KB modem. This was a receiver problem related to excessive wideband digital noise getting into the receiver RF stages. This will be resolved before production.

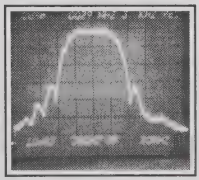
Performance with off frequency signals seems to be at least as good as the original design, degrading only about 1 db with a 5 KHz frequency offset.

The response time from RTS to DCD has been measured at about 3 milliseconds. I'm using a TXDELAY value in NOS of 5 ms and haven't encountered any problems. This is much faster than the original design which required a TXDELAY setting of 15 ms.

The transmitter spectral bandwidth is about the same as the original design.



Transmited spectrum
Horizontal: 50 KHz/division
Vertical: 10 DB/division



Transmited spectrum
Horizontal: 20 KHz/division
Vertical: 10 DB/division

Applications

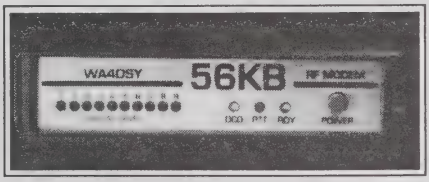
This modem can find uses in several areas. When the original modem was introduced in 1987, the computing power available to the average ham was quite limited and had problems keeping up with 56KB data. Today (1995) the average ham can afford a 66 MHz 486 machine. Multitasking operating systems such as OS/2 and Linux running on these machines allow hams to set up their own TCPIP Web sites on the air. Applications such as a Web server are useless at 1200 baud. For this reason, I believe this modem has as much potential for use as a user LAN modem as it does for point-to-point links. Assuming the built in full duplex bit repeater works as expected, I hope to see many 56KB FDX user LANs spring up around the world. They would work just like an FM voice repeater except for the 70 KHz bandwidth requirment. There is one such LAN in Ottawa, Ontario, Canada.

We plan to put up a full duplex 56KB Metropolitan Area Network on 222.400 (input) and 223.85 MHz (output) here in the Atlanta, Georgia area using a 56KB modem, a receive converter, transverter and a Sinclair duplexer. Users, of course, only need a modem and transverter.

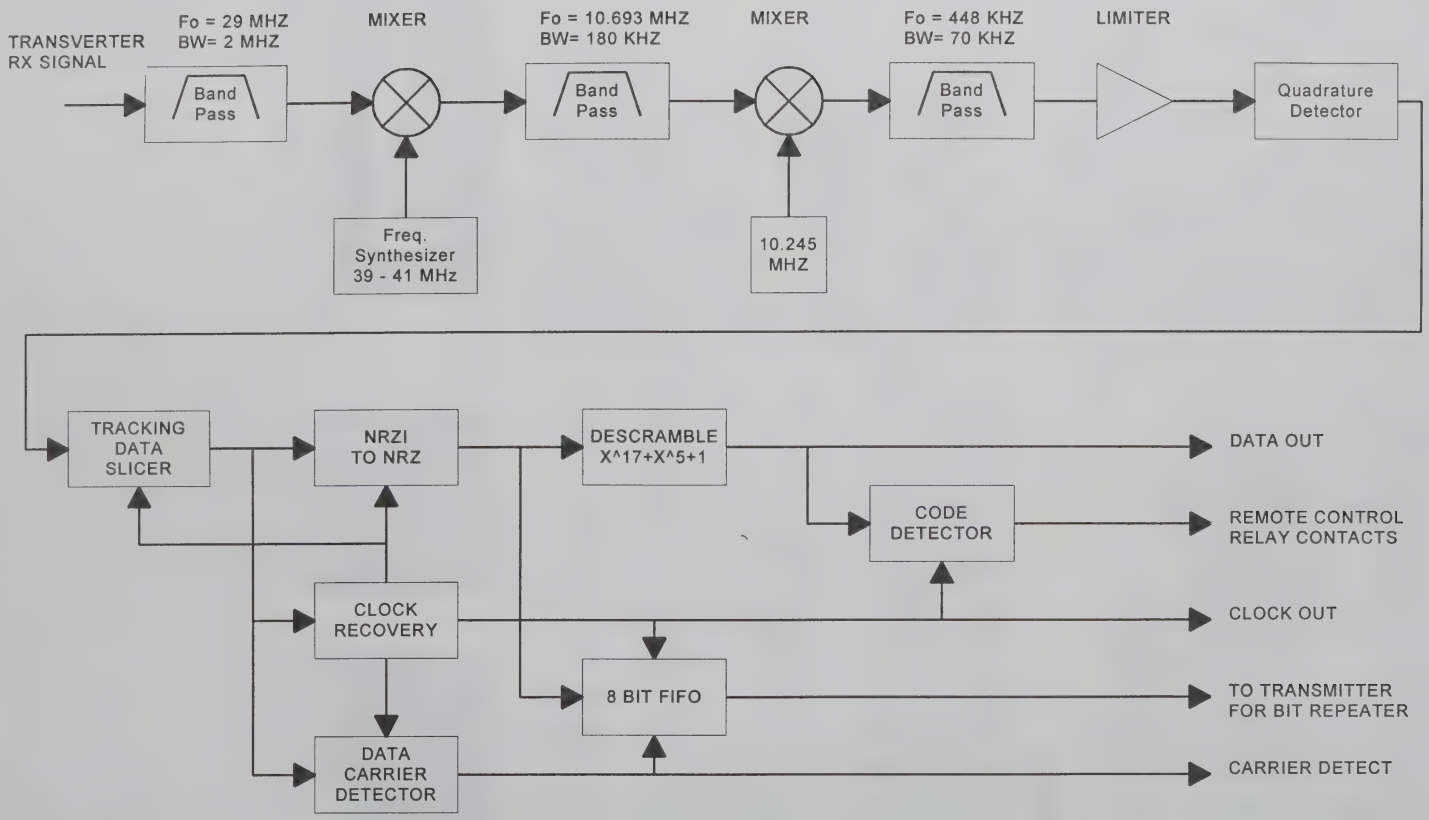
The U.S. now has a 1 MHz wide band (219 to 220 MHz) for "point-to-point fixed digital message forwarding". The band is divided into ten 100 KHz channels. This modem is ideal for that service.

Sales and Marketing

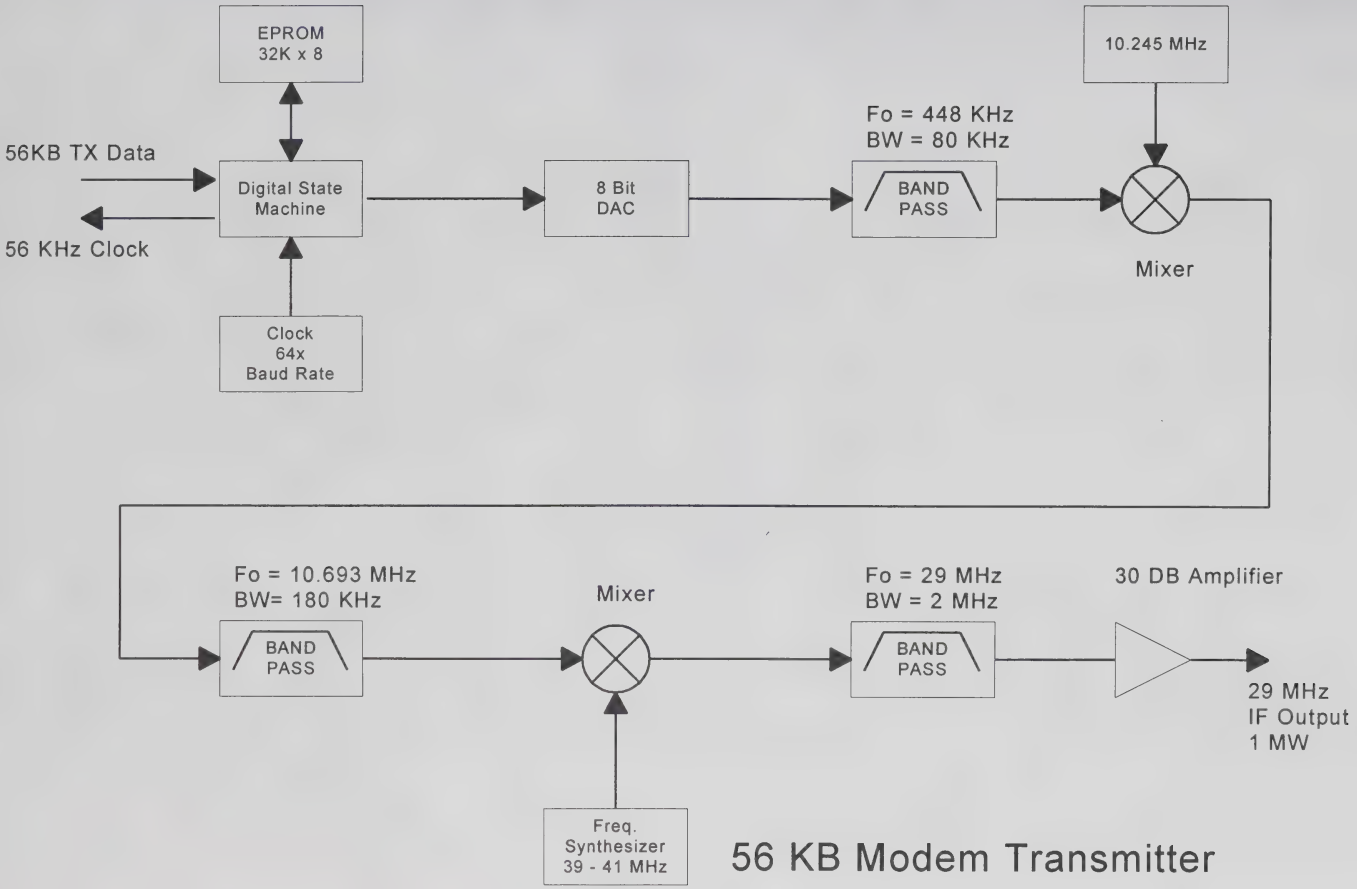
By now you're probably wondering how to get one of these modems. You can't, at least not right now. The modem is still in development. However, I'm negotiating with a well known manufacturer of packet radio equipment to produce and sell this modem. I hope to see them advertised for sale late in 1995 or early 1996. GRAPES will also be involved in modem sales.



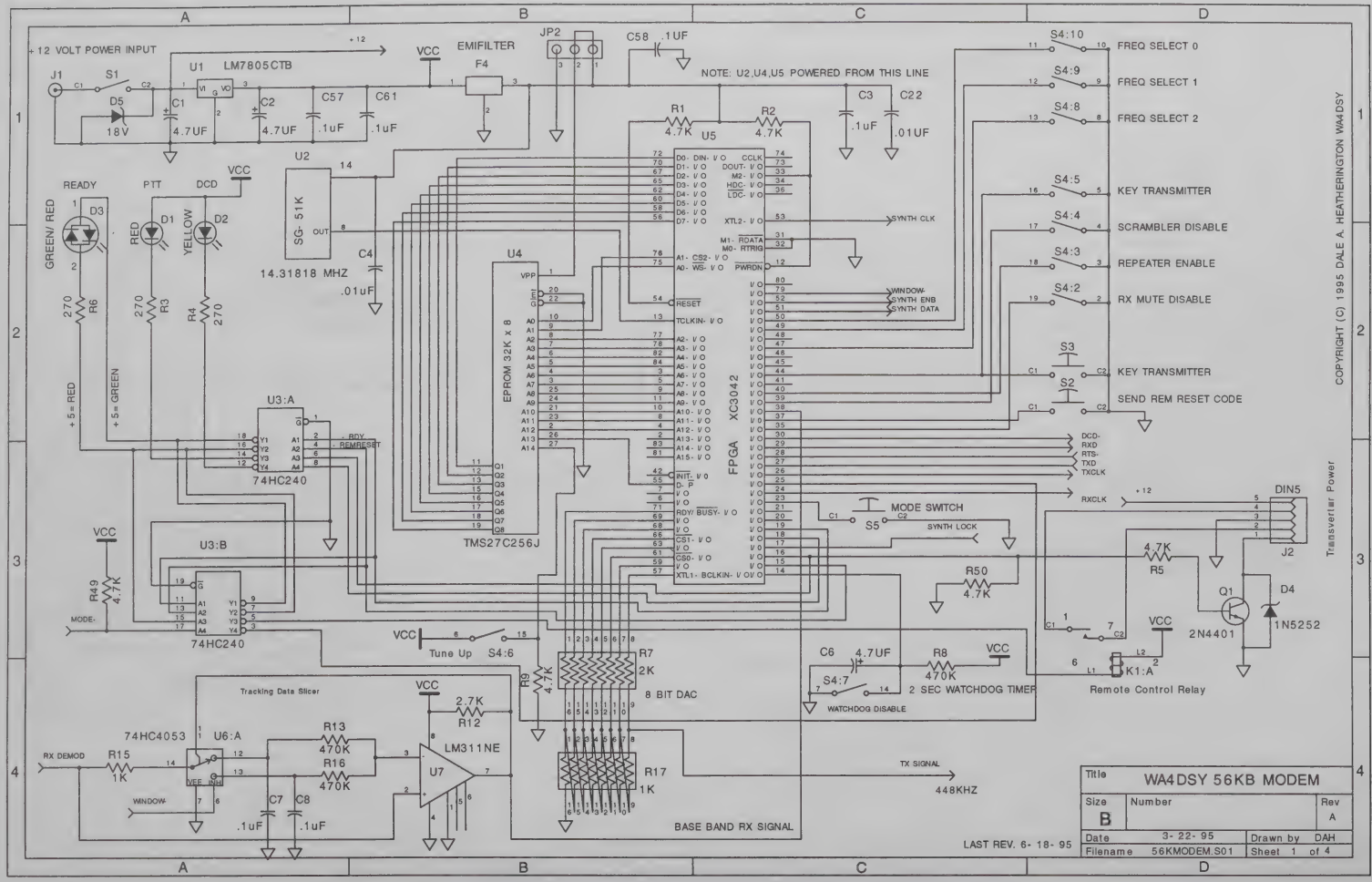
A Prototype Modem

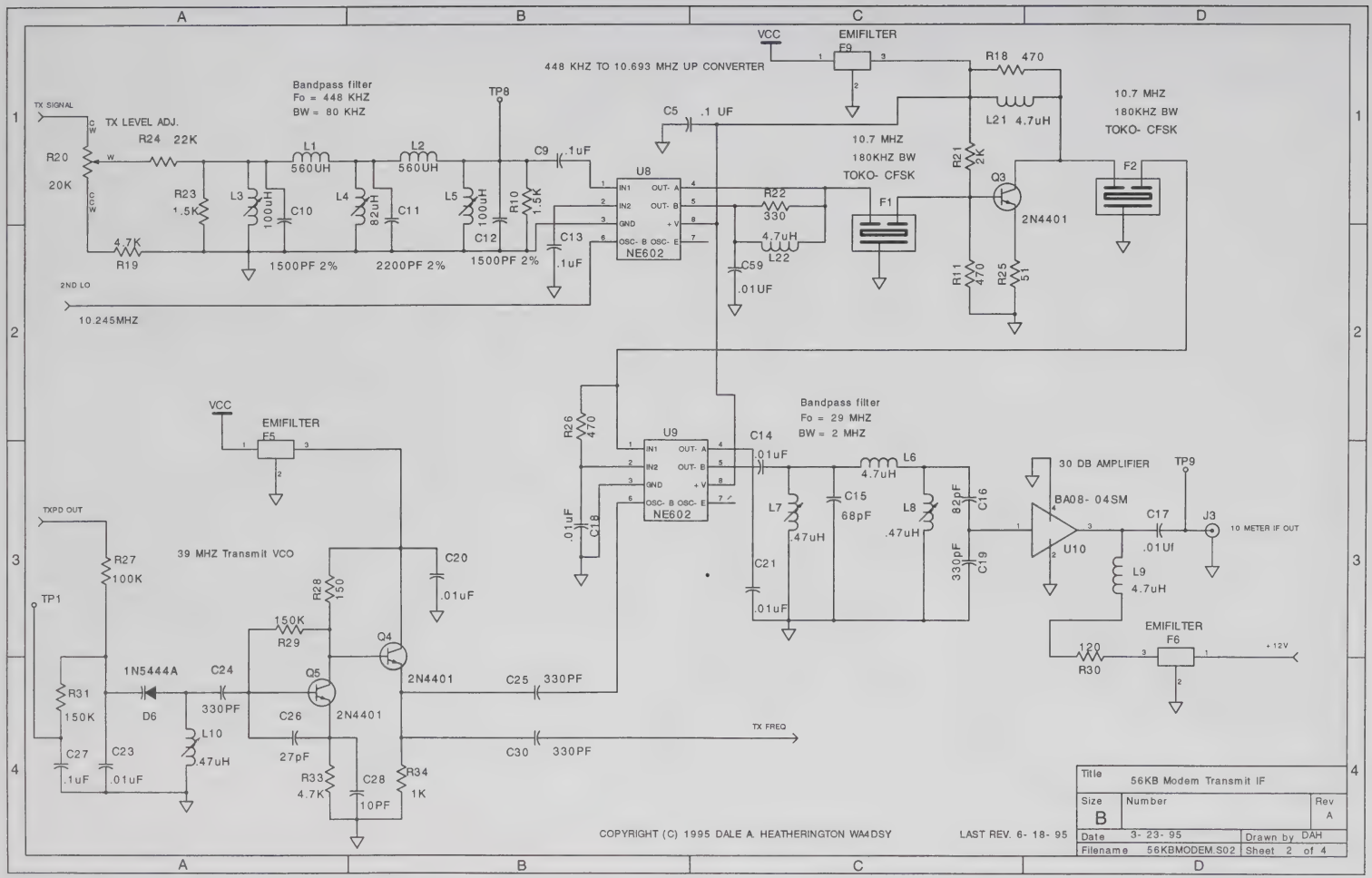


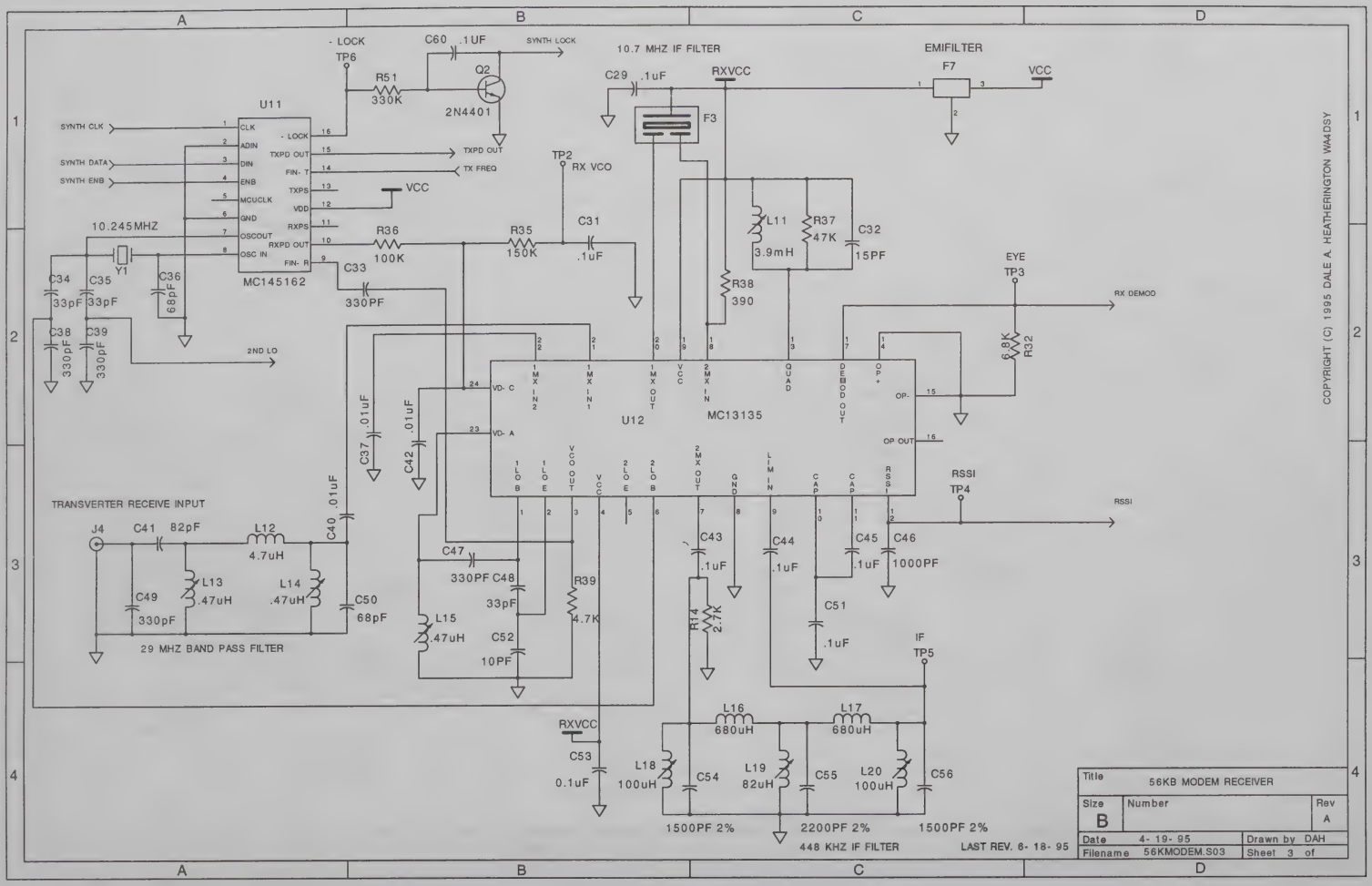
56 KB Modem Receiver Block Diagram

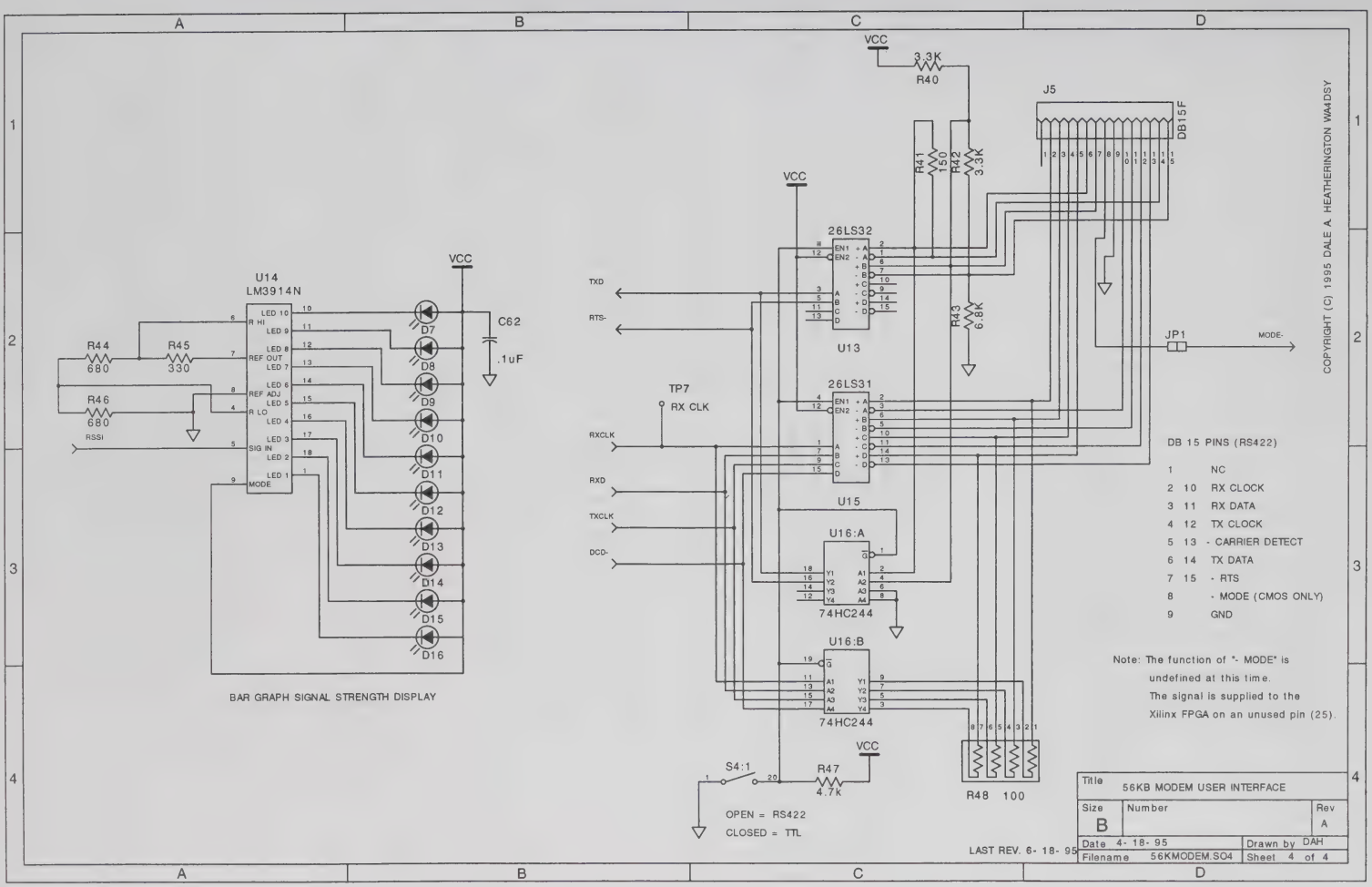


56 KB Modem Transmitter
Block Diagram


+ 12 VOLT POWER INPUT
U1 LM7805CTB
VCC
EMIFILTER
F4
JP2
C58 .1UF
NOTE: U2,U4,U5 POWERED FROM THIS LINE
J1
S1
D5
18V
C1 4.7UF
C2 4.7UF
C57 .1uF
C61 .1uF
R1 4.7K
R2 4.7K
C3 .1uF
C22 .01UF
U2
SG- 51K
14.31818 MHZ
C4 .01uF
U5
FPGA XC3042
U4
EPROM 32K x 8
TMS27C256J
SYNTH CLK
WINDOW-
SYNTH ENB
SYNTH DATA
READY
PTT
DCD
GREEN/ RED
RED
YELLOW
D3
D1
D2
R6 270
R3 270
R4 270
+ 5= RED
+ 5= GREEN
U3:A
74HC240
- RDY
- REMRESET
U3:B
74HC240
R49 4.7K
MODE-
Tune Up S4:6
R7 2K
8 BIT DAC
R9 4.7K
R17 1K
BASE BAND RX SIGNAL
Tracking Data Slicer
74HC4053
U6:A
RX DEMOD
R15 1K
R13 470K
R16 470K
C7 .1uF
C8 .1uF
WINDOW-
R12 2.7K
LM311NE
U7
TX SIGNAL
448KHZ
MODE SWITCH
S5
SYNTH LOCK
R50 4.7K
C6 4.7UF
R8 470K
S4:7
WATCHDOG DISABLE
2 SEC WATCHDOG TIMER
S4:10
FREQ SELECT 0
S4:9
FREQ SELECT 1
S4:8
FREQ SELECT 2
S4:5
KEY TRANSMITTER
S4:4
SCRAMBLER DISABLE
S4:3
REPEATER ENABLE
S4:2
RX MUTE DISABLE
S3
KEY TRANSMITTER
S2
SEND REM RESET CODE
DCD-
RXD
RTS-
TXD
TXCLK
RXCLK
DIN5
J2
Transverter Power
R5 4.7K
Q1
2N4401
D4
1N5252
K1:A
Remote Control Relay
COPYRIGHT (C) 1995 DALE A. HEATHERINGTON WA4DSY
Title WA4DSY 56KB MODEM
Size B
Number
Rev A
Date 3- 22- 95
Drawn by DAH
Filename 56KMODEM.S01
Sheet 1 of 4
LAST REV. 6- 18- 95


448 KHZ TO 10.693 MHZ UP CONVERTER
VCC
EMIFILTER
F9
R18 470
L21 4.7uH
10.7 MHZ
180KHZ BW
TOKO- CFSK
TX SIGNAL
TX LEVEL ADJ.
R24 22K
R20
20K
Bandpass filter
Fo = 448 KHZ
BW = 80 KHZ
TP8
C5 .1 UF
L1
560UH
L2
560UH
C9 .1uF
U8
IN1
IN2
GND
OUT- A
OUT- B
+ V
OSC- B OSC- E
NE602
R22
330
4.7uH
L22
C59
.01UF
F1
F2
Q3
2N4401
R21
2K
R11
470
R25
51
R23
1.5K
L3
100uH
C10
L4
82uH
C11
L5
100uH
C12
R10
1.5K
C13
.1uF
1500PF 2%
2200PF 2%
1500PF 2%
R19
4.7K
2ND LO
10.245MHZ
VCC
EMIFILTER
F5
Bandpass filter
Fo = 29 MHZ
BW = 2 MHZ
U9
R26
470
C14
.01uF
L6
4.7uH
30 DB AMPLIFIER
TP9
BA08- 04SM
U10
C17
.01Uf
J3
10 METER IF OUT
L9
4.7uH
C15
68pF
L7
.47uH
L8
.47uH
C16
82pF
C19
330pF
C21
.01uF
C18
.01uF
TXPD OUT
R27
100K
TP1
39 MHZ Transmit VCO
C20
.01uF
R28
150
150K
R29
Q4
2N4401
1N5444A
C24
Q5
2N4401
C25 330PF
R31
150K
D6
330PF
C26
L10
.47uH
27pF
C27
.1uF
C23
.01uF
R33
4.7K
C28
10PF
R34
1K
C30 330PF
TX FREQ
EMIFILTER
F6
120
R30
+ 12V
COPYRIGHT (C) 1995 DALE A. HEATHERINGTON WA4DSY
LAST REV. 6- 18- 95
Title
56KB Modem Transmit IF
Size
B
Number
Rev
A
Date
3- 23- 95
Drawn by DAH
Filename
56KBMODEM.S02
Sheet 2 of 4


- LOCK
TP6
C60 .1UF
SYNTH LOCK
R51
330K
Q2
2N4401
10.7 MHZ IF FILTER
EMIFILTER
F7
VCC
C29 .1uF
RXVCC
F3
U11
SYNTH CLK
SYNTH DATA
SYNTH ENB
CLK
ADIN
DIN
ENB
MCUCLK
GND
OSCOUT
OSC IN
- LOCK
TXPD OUT
FIN- T
TXPS
VDD
RXPS
RXPD OUT
FIN- R
TXPD OUT
TX FREQ
VCC
TP2
RX VCO
10.245MHZ
Y1
C34 33pF
C35 33pF
C36 68pF
MC145162
R36 100K
R35 150K
C31 .1uF
C33 330PF
L11 3.9mH
R37 47K
C32 15PF
R38 390
EYE
TP3
RX DEMOD
6.8K R32
2ND LO
C38 330pF
C39 330pF
U12
MC13135
VD- C
VD- A
MIXIN2
MIXIN1
MIXOUT
VCC
2MIXIN
QUAD
DEMODOUT
OP+
OP-
OP OUT
C37 .01uF
C42 .01uF
RSSI
TP4
RSSI
TRANSVERTER RECEIVE INPUT
J4
C41 82pF
L12 4.7uH
C40 .01uF
C49 330pF
L13 .47uH
L14 .47uH
C50 68pF
29 MHZ BAND PASS FILTER
C47 330PF
C48 33pF
L15 .47uH
C52 10PF
R39 4.7K
C43 .1uF
R14 2.7K
C44 .1uF
C45 .1uF
C46 1000PF
C51 .1uF
IF
TP5
RXVCC
C53 0.1uF
L16 680uH
L17 680uH
L18 100uH
C54
L19 82uH
C55
L20 100uH
C56
1500PF 2%
2200PF 2%
1500PF 2%
448 KHZ IF FILTER
LAST REV. 6- 18- 95
COPYRIGHT (C) 1995 DALE A. HEATHERINGTON WA4DSY
Title 56KB MODEM RECEIVER
Size B
Number
Rev A
Date 4- 19- 95
Drawn by DAH
Filename 56KMODEM.S03
Sheet 3 of


U14
LM3914N
R44
680
R45
330
R46
680
RSSI
VCC
C62
.1uF
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
BAR GRAPH SIGNAL STRENGTH DISPLAY
VCC
3.3K
R40
J5
DB15F
R41
3.3K
R42
3.3K
26LS32
U13
TXD
RTS-
R43
6.8K
JP1
MODE-
TP7
RX CLK
26LS31
U15
RXCLK
RXD
TXCLK
DCD-
U16:A
74HC244
U16:B
74HC244
S4:1
R47
4.7K
R48 100
OPEN = RS422
CLOSED = TTL
DB 15 PINS (RS422)
1 NC
2 10 RX CLOCK
3 11 RX DATA
4 12 TX CLOCK
5 13 - CARRIER DETECT
6 14 TX DATA
7 15 - RTS
8 - MODE (CMOS ONLY)
9 GND
Note: The function of "- MODE" is undefined at this time. The signal is supplied to the Xilinx FPGA on an unused pin (25).
COPYRIGHT (C) 1995 DALE A. HEATHERINGTON WA4DSY
Title
56KB MODEM USER INTERFACE
Size
B
Number
Rev
A
Date 4- 18- 95
Drawn by DAH
Filename 56KMODEM.SO4
Sheet 4 of 4
LAST REV. 6- 18- 95

Extended sequence number (modulo-128) option for AX.25

Rob Janssen, PE1CHL

ABSTRACT

An extension to the AX.25 protocol is proposed, to enhance the efficiency of transmissions of large numbers of small packets on half-duplex interlinks. The sequence number space is increased from 8 to 128 to accomodate larger values of MAXFRAME, and procedures are described to enable monitoring of extended sequence number frames, and resequencing in case of frame loss.

1. Introduction

Half-duplex packet radio links are operated at ever higher bitrates. When observing the efficiency of a half-duplex link, it is immediately obvious that a percentage of the maximum throughput is lost due to delays when changing the direction of the traffic. A transmitter needs a finite time to key-up, and the receiver needs time to lock on the received signal and provide stable data. At the end of a transmission, some extra flags are usually sent to overcome a difficulty in the commonmy used SCC chip, and to clear a scrambler that may be present in the path. When the link would be via satellite, propagation delay would be an extra factor.

To operate at reasonable efficiency, it is best to send at least so much data in each transmission that the changeover delay is less than, say, 10% of the transmit time. This is normally equivalent to transmitting about 1..3 seconds.

On a link between nodes operating using the NET/ROM protocol, there is only a single AX.25 connection that handles all the traffic. On its queue are both the user-data packets and the transport layer acknowledgements, in addition to the transport layer connection setup packets. In practice, this means there are often quite a number of small packets queued on the connection.

To get the best effciency (on a link with a reasonable bit-error rate), one would like to send many of these packets in a single transmission. For example, when a number of 50-byte packets is queued and the target transmit time is 1 second, one would need to send about 24 packets at 9600 bps. However, the current AX.25 protocol uses only 3-bit sequence number fields, and therefore no more than 7 frames can be sent in a single transmission. This would be only a 300ms transmission in this case, which is often quite short when compared to the changeover delay.

To overcome this limitation, I have added the option of using extended (modulo-128) sequence numbers to the AX.25 handling in NETCHL over two years ago. It has operated very satisfactorily over that period, and this document describes how it was done, so that other software writers who desire to to the same can do it in a compatible way.

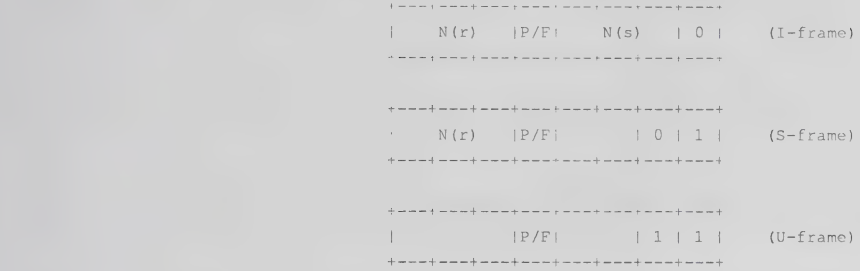
2. Protocol modifications

Basically, there is not much to it. The AX.25 protocol is based on HDLC, and in the HDLC standard there already exists the modulo-128 sequence number option. This essentially is what is being used.

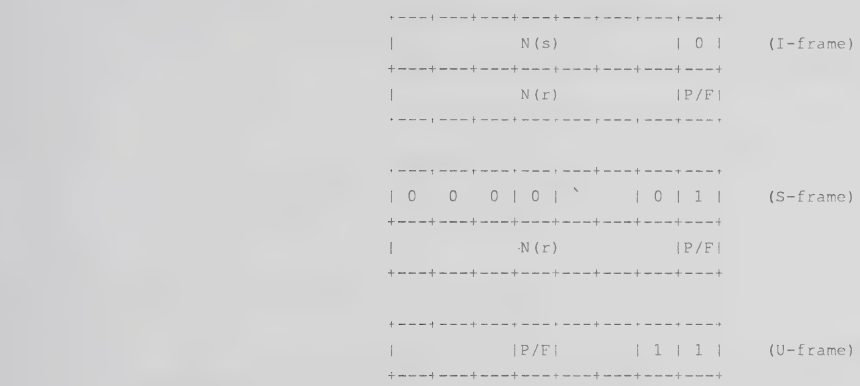
A extended sequence connection is started using SABME (0x6f) instead of SABM (0x2f). When the connected station answers the SABME with an UA, the connection is established and will use modulo-128 sequence numbers. This results in a change in the control field in each I and S frame, frames that include a sequence number. The control field becomes 16 bits long in these frames. U frames, including SABM, SABME, UA, DM, and UI are sent with the normal 8-bit control field.

(the bytes are shown MSB-to-the-left here)

modulo-8:

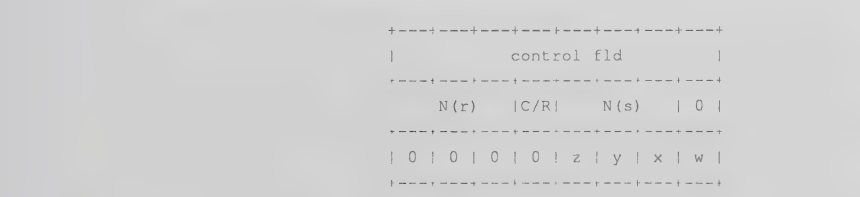


modulo-128:

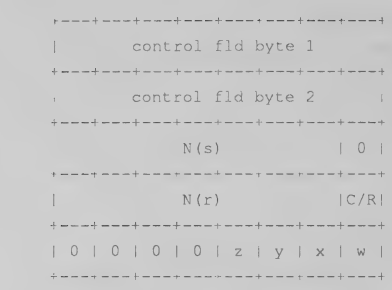


In the FRMR frame there is also a change in the data field of the frame:

modulo-8:



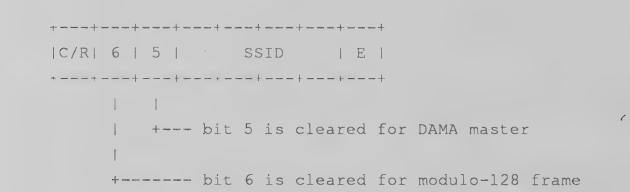
modulo-128:



When the rejected frame is a U-frame, the control field is put in the first byte of the FRMR frame, and the second byte is set to zero.

3. Monitoring

As the type of the connection (modulo-8 or modulo-128) is determined only at the start, by sending SABME instead of SABM, it is difficult for monitoring stations to know how to decode a received frame. Therefore, bit number 6 in the SSID field of the SOURCE call in the AX.25 header is cleared to indicate that the frame belongs to a modulo-128 connection. This is normally a reserved bit that should be set to 1. The value of bit 6 is not used by the stations participating in the connection, but is only intended as information for decoding by a station tracing (monitoring) the connection.



4. Implementation

Implementation of the extended sequence number (modulo-128) option should not be difficult when the existing AX.25 handler is reasonably well written.

- For each connection, an extra flag is required to indicate that modulo-128 sequence numbering is in use. The flag is set when SABME (0x6f) was received to setup the connection instead of SABM (0x2f).
- To setup a connection, SABME is sent when modulo-128 is desired, and the modulo-128 flag is set as well.
- The extraction of N(r), N(s) and P/F from the control fields of I and S frames must be made dependent on the setting of the modulo-128 flag.
- When constructing an I or S frame, the format must be selected depending on the modulo-128 flag.
- When incrementing a sequence number, the correct modulo must be selected depending on the flag.
- When constructing or interpreting a FRMR frame, the format of the frame is selected depending on the flag.
- Monitor (trace) code should be adapted to trace extended sequence number frames when bit 6 of the source SSID field is zero.
- A separate MAXFRAME value for modulo-128 connections ('EMAXFRAME') could be a settable parameter, when two types of connections are allowed on the same interface (port).
- The worst-case frame length (256-byte frame sent via 8 digipeaters) is one byte more than in modulo-8 mode. This will usually be no problem, as modulo-128 is most likely to be used on point-to-point links. For a really complete implementation, it may be that the buffer size has to be increased.

In (NETCHL) practice it has turned out to be convenient to use a variable in the connection control block, mmask, which is set to 7 for modulo-8 connections and to 127 for modulo-128. This variable is used as the "modulo-128" flag described above, and also as the "modulo mask" which is applied (ANDed) after all sequence number arithmetic.

5. Resequencing

It is obvious that, when sending so many frames in a single transmission, any lost frame has a large impact on throughput. In the standard AX.25 protocol, all frames following the missing frame have to be discarded and re-sent after the lost frame has been re-sent and successfully received.

It has been shown before that a resequencing queue ("framesammler") can be used to save the out-of-sequence frames, and use them after the lost frame has been retransmitted. Unfortuately, special tricks (checksumming the frame) were required to solve the ambiguity that results from the small sequence number range in standard AX.25.

With extended sequence connections, resequencing is possible without these tricks, when the EMAXFRAME (maximum number of unacknowledged frames) is kept below half the modulo. Therefore it is hereby specified that the value set for EMAXFRAME should be limited to 63. Then, for each incoming frame it can be unambiguously determined if it is a re-transmitted old frame or a frame which lies beyond the expected sequence number, and could be saved in a resequencing queue.

Resequencing can work without further extending the protocol (adding a Selective REJect frame type), using the following simple rules:

- when a frame with a sequence number beyond the expected number is received, it is saved on a resequencing queue for the connection. (unless it is already there, or memory is low)
- each time a frame has been received that matches the expected sequence number, the resequencing queue is examined to see if it contains one or more frames that fit in next
- when a reply is to be sent to report the next expected sequence number, the following type of frame is sent:
  RNR when the input queue is too long
  REJ when frames are present on the resequencing queue
  RR otherwise
- when a REJ reply is received, the next transmission will be only a single frame, namely the next expected frame at the other end.

This procedure is not as efficient as SREJ would be, but it is in successfull operation in a number of programs that implement resequencing. It results in a short transmission to recover from loss of a frame, but does not waste time retransmitting information the other end already has.

6. Compatability

NET implements compatability with old code using the following algorithm: For each connection configured to be in modulo-128 mode, SABME is sent first. When the response is DM or FRMR, a fallback is done to modulo-8 mode and SABM is sent instead. Unfortunately it turns out that not all AX.25 implementations send DM or FRMR back when they receive SABME (not even with Poll bit set). This can be regarded as a violation of the protocol, but on the other hand sending an SABME could be regarded as a protocol violation as well, so... The solution has been to have a table of callsigns that are known to handle modulo-128 connections, and only attempt such a connection when the destination station is in the table.

On NET/ROM interlinks, which is where it is most useful, I think it should be sufficient to have a single configuration bit per interface (port) enabling use of modulo-128. It is also no problem to implement it on incoming connections on local access interfaces. When it is desired to have modulo-128 operation on outgoing connects, on interfaces where many stations are likely to be present, a table of callsigns is required for compatability.

Rob PE1CHL - Mar 4, 1995

DSP-93 Update: The TAPR/AMSAT Joint DSP Project

Greg Jones, WD5IVD (wd5ivd@tapr.org)
Bob Stricklin, N5BRG (n5brg@tapr.org)
Robert Diersing, N5AHD (n5ahd@amsat.org)

Tucson Amateur Packet Radio, Corp.
8987-309 E. Tanque Verde Rd #337
Tucson, AZ 85749-9399
tapr@tapr.org • http://www.tapr.org/tapr

Abstract

This paper discusses the geniuses of the TAPR/AMSAT Joint DSP project. The end result was the TAPR/AMSAT DSP-93. As of July 1995, 300 units have been made available to amateurs for construction. Items to be covered include: a history of the project, history of the DSP-93, DSP-93 design, current software suite, code development, and comments on the successful use of Internet for project support.

Introduction

In 1987, DSP-based modems were seen as the coming amateur trend [1,2]. Digital signal processing techniques have been a promise of improved reliability and lower cost since 1986. In July 1988, TAPR and AMSAT entered into the Joint DSP Program, in order to fund the development of an eventual DSP unit for amateur usage. The hope was to fulfill the dream of having a low-cost, flexible amateur kit. The DSP-1 was based on a TI TMS320C15 at 25Mhz, with 4K of program memory. In addition, a GPP board (NEC V40 with 72001 serial I/O chip) was developed to interface the DSP-1 to a PC for programming. Progress was made during the early stages of the project [3,4], but later trailed off as other projects appeared. The articles in the reference provide a good snapshot for those early days. The DSP-1 platform, along with experience gained from the earlier Delanco Spry boards, while never reaching a final kit, proved invaluable in the learning process. These early developments are reflected to some degree in the final DSP-93 project.

In 1993, the TAPR/AMSAT co-management team made the deliberate decision to stop work on the DSP-1 project and to proceed with a proposal by Bob Stricklin, N5BRG, in Dallas, Texas that focused on a modular, stand alone DSP system. The Stricklin KISS design [5], later renamed DSP-93, offered many of the elements that the initial design goals of the project had specified. Seven years into the joint project and fifteen months after the decision was made to proceed with the DSP-93 development, TAPR and AMSAT shipped the first 110 units of the project. In July of 1995, a second batch of 180 units was produced and shipping began — an end to a long journey.

History

The history of DSP-93 can be traced back to 1991 when development began on the KISS DSP. The KISS DSP was a single board DSP engine designed around the Texas Instruments TMS320C25. The KISS DSP board, as described in an earlier AMSAT publication [5], included the DSP, memory, serial I/O, and a limited audio interface. The KISS DSP was designed as a stand alone box to interface between the computer and radio. In some cases, a TNC was needed to process the HDLC digital data. To simplify the computer interface, a monitor routine (DSP BIOS) was developed which included functions to download DSP programs from a computer, view and change program memory, data memory, and DSP registers. To make software development possible, work was undertaken with Thomas Anderson of Speech Technology in Issaquah, Washington to produce a TMS320C25 assembly table for his assembler. The result was an inexpensive assembler for the a TMS320C25-based DSP.

Several amateurs obtained and built the KISS DSP boards and placed them in operation. The KISS DSP experiments served as an excellent initial platform to develop and understand the realities of developing DSP software and hardware for amateur applications. Areas which needed work were the radio interface circuitry and understanding how to match the hardware with the particular application. A single 'do all' DSP box is not a practical solution to all amateur DSP applications. It is not "Only Software." It is a combination of selecting the best hardware for the particular application and developing the software around the selected hardware. Since most amateurs are looking for a cost effective solution, you can sacrifice some of the capability and provide them as future additions.

TAPR/AMSAT DSP-93

Based on the KISS DSP lessons, DSP-93 was designed in a modular fashion with multiple four-layer boards which include the interconnecting bus structure. The boards include a DSP engine board and a radio and computer interface board. Any of these boards can be replaced with a future board designed for any number of unique applications. It's sort of like adding a new application card to a PC without redesigning the complete PC. Figure 1 shows the basic hardware black diagram. Figure 2 shows overall structure of the assembled unit with enclosure.

The bottom board (DSP Engine) contains the TMS320C25 DSP, 64K by 16 bits of program and data memory, the clock circuitry and programmable array logic (PALS) for system I/O. All the DSP lines are connected to the backplane bus. This was intended to make it easier to add additional features on additional DSP-93 boards. The floating input lines all have 100K pull up resistors. The clock circuit for driving the DSP is also included on the DSP Engine board. Since slower EPROMs are used to boot the system, the clock is designed to shift between half the maximum rate and full speed. Clock shifts are software controlled and will be transparent to the application. The target clock speed is 40 MHz; however, we have demonstrated that 26 MHz is adequate for most amateur applications. Additional testing of the Beta boards will be required to establish the final clock rate.

The top board (Radio Interface Board) contains two eight pin mini-DIN connectors for the radio interface. Incoming radio signals pass through a voltage divider to establish the initial levels, then through an eight channel multiplex chip. The multiplex

Figure 1: Basic hardware black diagram.

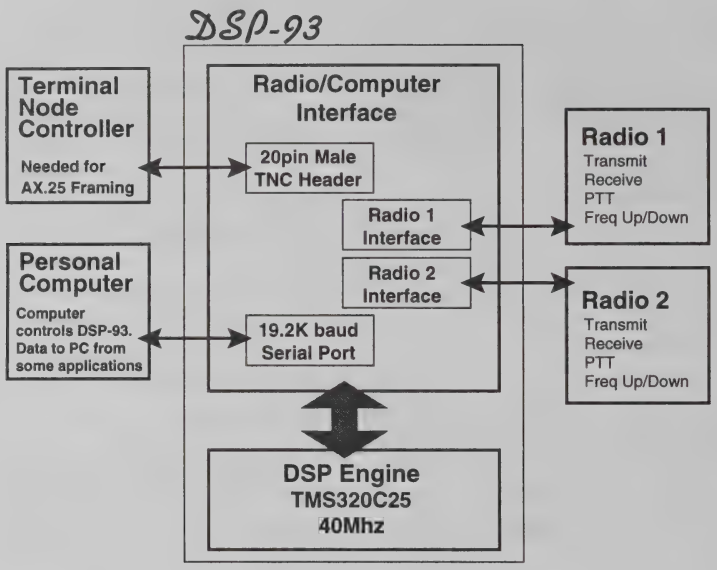
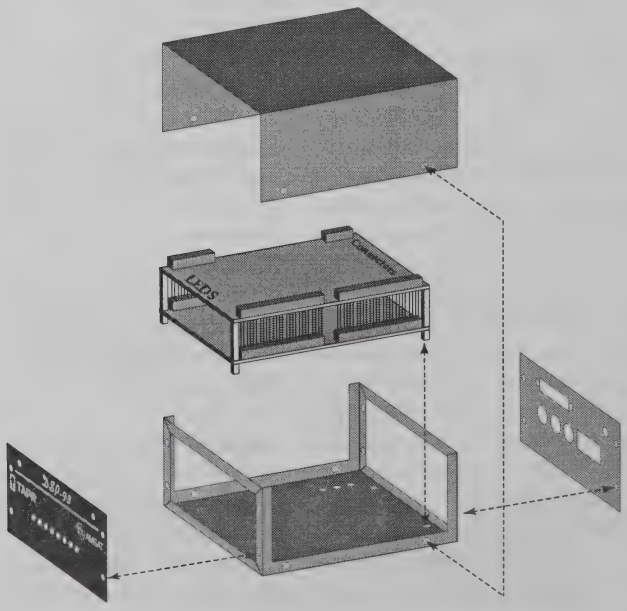


Figure 2: Overall structure of the assembled unit with enclosure.



chip will feed the single A/D input with either of the radio inputs or one of the six auxiliary inputs. Before reaching the A/D, the signal is conditioned by an OP-AMP whose gain is software selectable. Gain options include; 1X, 2X, 5X, 10X, 20X, 100X, 1000X and a comparator mode with ground being the reference voltage. Once again this is all software controlled and it should make for some nice adaptive signal conditioning algorithms. The Texas Instruments TLC32047 Analog I/O chip is used on board two. This chip samples and updates at a rate of 45K operations per second and it includes aliasing filters. The chip communicates with the DSP using a 5 MHz serial interface. The 5 MHz interface is routed through a selection PAL to allow the user to disable this serial device in favor of another.

Code Development

A low cost shareware assembler is available for code development. Thomas Anderson of Speech Technology in Issaquah, Washington has worked with the DSP-92 development group to produce a TMS320C25 assembly table for his assembler. All the details needed to write DSP code is supplied with the kits. An in-depth programming guide has been developed and is available for DSP-93 developers.

Project Team

Much of the development and current success of the DSP-93 project can be contributed to the designers, developers, and testers. Designer: Bob Stricklin, N5BRG. Project Managers: Bob Stricklin, N5BRG and Greg Jones, WD5IVD. Joint DSP Project Officers: Robert Diersing, N5AHD (AMSAT) and Greg Jones, WD5IVD (TAPR). The Alpha-Team: Bob Stricklin, N5BRG, Frank Perkins, WB5IPM, Jon Bloom, KE3Z, Lon Cecil, WB5PKJ, Tom McDermott, N5EG, Robert Diersing, N5AHD, UoSAT/ Doug Loughmiller, K05I/G0SYX, John Conner, WD0FHG, Greg Jones, WD5IVD, and Bill Reed, WD0ETZ. The Beta-Team: Jack Davis, WA4EJR, Paul Beckmann, WA0RSE, Scott Zehr, K9GKC, Ron Parsons, W5RKN, Jim Tittsler, 7J1AJH/AI8A, Michael Zingman, N4IRR, Stan Salek, KD6CVL, Mark Hammond, KC4EBR, Marcel Losekoot, Bill Beech, NJ7P, Gould Smith, WA4SXM, Roy Welch, W0SL, Greg Ratcliff, NZ8R, Brian Straup, NQ9Q, Doug Howard, KG5OA, and Robert Greenfield, VE3DSC.

The current DSP-93 development group consists of: Bob Stricklin, N5BRG, Frank Perkins, WB5IPM, Tom McDermott, N5EG, Ron Parsons, W5RKN, Scott Zehr, K9GKC, Stan Salek, KD6CVL, Mark Hammond, KC4EBR, Gould Smith, WA4SXM, Roy Welch, W0SL, Brian Straup, NQ9Q, Michael Zingman, N4IRR, Lon Cecil, WB5PKJ, and Robert Diersing, N5AHD.

Basic Software Suite

As of June 1995, the DSP-93 had the following software available. Some modems have multiple versions to accommodate various radios and are not shown.

9600 bps Modems		
FSK93U1/2	Half or full duplex operation, active DCD	WB5IPM/W5RKN/W0SL
FSK93U1/2B	Weak signal optimized FSK93U1/2	WB5IPM/W5RKN/W0SL
FSK93K1/2	ISI comp. for KO-23/25, active DCD	WB5IPM/W5RKN/W0SL
TPRS931/2	TPRS with NRZI in/out, active DCD	WB5IPM
1200 bps Modems		
PKT931/2	Standard AFSK, active DCD	WB5IPM
PSK93R1/2J	Satellite PSK, active DCD, SmartTune	WB5IPM
PSK93R1/2Y	Sat. PSK, active DCD, SmartTune, YAESU	WB5IPM
HF Modems		
HOT_HF93	RTTY modem, Viterbi soft detection	W3HCF
HF_93HT1/2	300 bps FSK, active DCD, adpt. threshold	WB5IPM
HF_93U1/2	AMTOR using TOR.EXE, adpt. threshold	WB5IPM
Sat. Pic/Telem		
APT_93HX1/2	Carrier-sync. pixels eliminate doppler bow	WB5IPM
P3C93T1/2	AO-13 400 bps telemetry, P3C.EXE comp.	WB5IPM
UO11	UO-11 telemetry decode.	WB5IPM
Noise Processing		
W9GR_93	Adpt. carrier null, channel enh., BP filters	N4IIR/K9GKC/W9GR
CW93_D	CW filter	WB5IPM
Instrumentation		
D93WE (PC)	Control Loader, audio oscilloscope, function generator and spectrum analyzer (Figure 3)	N5EG
DSP-93 (Mac)	Control Loader, audio oscilloscope (Figure 4), function generator and spectrum analyzer	W5RKN
Diagnostics		
Monitor	Basic DSP-93 Monitor and SW utilities	N5BRG
DSP-93 tests	Check-out of all DSP-93 functions	WB5PKJ/N5EG/W5RKN

Software is being distributed on Internet, the Amateur Satellites, as well as being made available on disk as part of the TAPR software library. The idea of software for the DSP-93 is to make it as easy as possible to get and upgrade software in the future.

Figure 3 - Spectrum Analyzer Display from DSP93WE Windows [10]

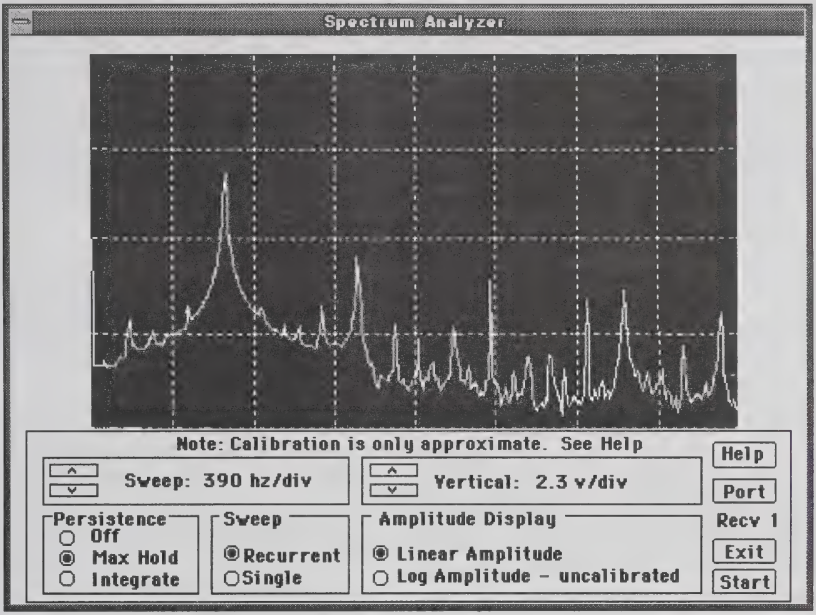
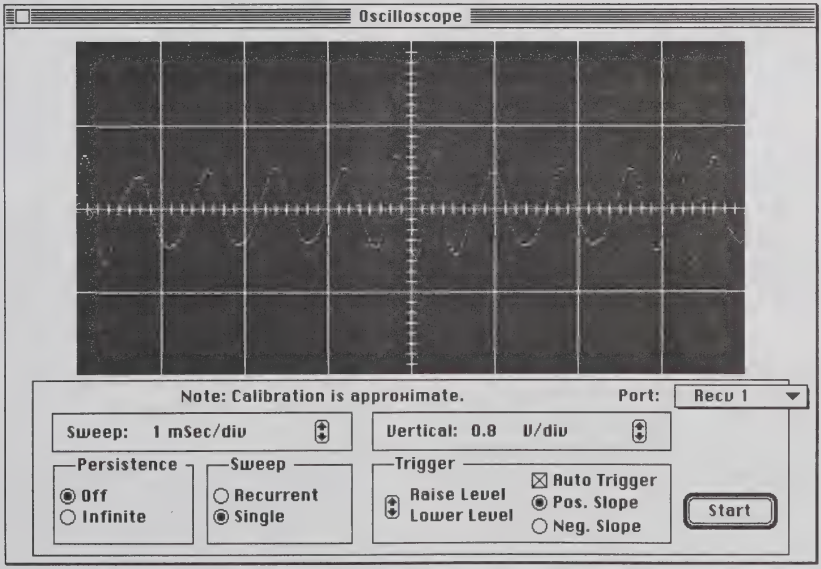


Figure 4 - Sampling Oscilloscope from DSP93 Mac [11]



Oscilloscope: The display shows a 1200 bps packet audio signal. Note the two different periods of the waveform corresponding to different binary digits.

Success of Internet Support

The project, from the beginning, has been supported using only the Internet. Internet and satellite communications have been a major key to the success of the project. Internet e-mail allowed one of the few TAPR projects to include small groups of individuals outside of the main development group located in the Dallas area. We felt that the development group still had to be in a local geographical area or the development process would have been retarded. Much of the initial development required personal visits during the prototyping and initial alpha testing.

TAPR maintains an ftp site at ftp.tapr.org. Updates and archive material relevant to the DSP-93 can be obtained via anonymous ftp from the directory /tapr/dsp93. Copies of all past messages distributed on the dsp-93@tapr.org list by sending mail to 'listserv@tapr.org', subject of 'list', message of 'index dsp-93'. Recently a web page has been created with links to manuals, software, and quicktime movies of various modes. http://www.tapr.org/tapr

---

References:

1. McGwier, Robert, N4HY. DSP Modems: It's Only Software. (1987). Proceedings of the 6th ARRL Computer Networking Conference. Redondo Beach, California. August, 1987. ARRL: Newington, CT. p. 123.

2. Clark, Tom, W3IWI, and Robert McGwier, N4HY. (1987). Digital Signal Processing and Amateur Radio. Proceedings of the 6th ARRL Computer Networking Conference. Redondo Beach, California. August, 1987. ARRL: Newington, CT. p. 44.

3. Clark, Tom, W3IWI, and Robert McGwier, N4HY. (1988). The DSP Project Update. Proceedings of the 7th ARRL Computer Networking Conference. Columbia, Maryland. October, 1988. ARRL: Newington, CT. p. 48.

4. Johnson, Lyle, WA7GXD. (1988). The AMSAT/TAPR DSP 1 Project: Hardware Design. Proceedings of the 7th ARRL Computer Networking Conference. Columbia, Maryland. October, 1988. ARRL: Newington, CT. p. 101.

5. Stricklin, Bob, N5BRG. (1990). KISS DSP for Amateur Applications. Proceedings of the 8th AMSAT-NA Space Symposium. Houston, Texas. October 1990. AMSAT: Sliver Spring, MD. p. 22.

6. Stricklin, Bob and Greg Jones. (1993). TAPR/AMSAT DSP-93 Project. Proceedings of the 1993 AMSAT-NA. AMSAT.

7. Stricklin, Bob. (1993). DSP-93: The Joint DSP Program (TAPR/AMSAT). Issue #52, Fall 1993, Packet Status Register. pp. 4-5. Tucson Amateur Packet Radio Corp.

8. Stricklin, Bob. (1994). TAPR/AMSAT Joint DSP Project: DSP-93. Proceedings of the TAPR 1994 Annual Meeting. Tucson Amateur Packet Radio Corp.

9. DSP-93: The TAPR/AMSAT Joint DSP Program. (1994). Issue #55, Summer 1994, Packet Status Register. pp. 1-4. Tucson Amateur Packet Radio Corp.

10. McDermott, Tom, N5EG. (1995). D93WE Windows Development Environment for the TAPR/AMSAT DSP-93. Proceedings of the 1995 TAPR Annual Meeting. St Louis, MO. March, 1995. TAPR: Tucson, AZ. p. 31.

11. Parsons, Ron, W5RKN. (1995). DSP-93 Control Macintosh Development Environment for the TAPR/AMSAT DSP-93. Proceedings of the 1995 TAPR Annual Meeting. St Louis, MO. March, 1995. TAPR: Tucson, AZ. p. 38.

12. Parsons, Ron, W5RKN. (1995). Programming Guide for the DSP-93. June, 1995. TAPR: Tucson, AZ.

Introduction to Programming the TAPR/AMSAT DSP-93

Ron Parsons, W5RKN (w5rkn@amsat.org)
Don Haselwood, K4JPJ
Bob Stricklin, N5BRG (n5brg@tapr.org)

Tucson Amateur Packet Radio, Corp.
8987-309 E. Tanque Verde Rd #337
Tucson, AZ 85749-9399
tapr@tapr.org • http://www.tapr.org/tapr

Abstract

The purpose of this paper is to give a brief overview to assist potential programmers, new to the TAPR/AMSAT DSP-93 environment, insight into the tools and techniques available when developing for the DSP-93. The full developers/programming guide is available from TAPR [1].

Introduction

Any introduction regarding DSP-93 programming must begin with discussing how to locate and secure reference materials. The first step is to locate your local Texas Instruments distributor and call them. Local distributors have been known to give free access to their literature room. Books that you should be looking for include:

- **TMS320C2x User's Guide By Texas Instruments; Document # SPRU014C**
  This book covers the TMS320C25 DSP chip used in the DSP-93. It covers the chip's electrical properties, memory models, interrupt processing, and, of course, the instruction set.

- **Linear Circuits; Data Conversion, DSP Analog Interface, and Video Interface; Data Book Volume 2 By Texas Instruments; Document # SLYD004A.**
  This book covers the TLC32044CN AIO chip used in the DSP-93. It covers the chip's electrical properties, configuration, etc.

- **Digital Signal Processing Applications with the TMS320 Family; Theory, Algorithms, and Implementations**
  Volume 1 Document # SPRA012A
  Volume 2 Document # SPRA016
  Volume 3 Document # SPRA017
  These books cover various DSP algorithms which may or may not be useful to you.

- **Digital signal processing with the TMS320C25** Chassaing, Rulph, published 1990 by Wiley, New York 464 pg
  ISBN 0471510661

- **Digital Signal Processing: A laboratory approach using PC-DSP.** Alkin, Oktay, published 1994 by Prentice-Hall, Inc.
  ISBN 0-13-328139-6
  An introduction to Digital Signal Processing techniques. Comes with a DOS program which can compute FIR coefficients, among other things..

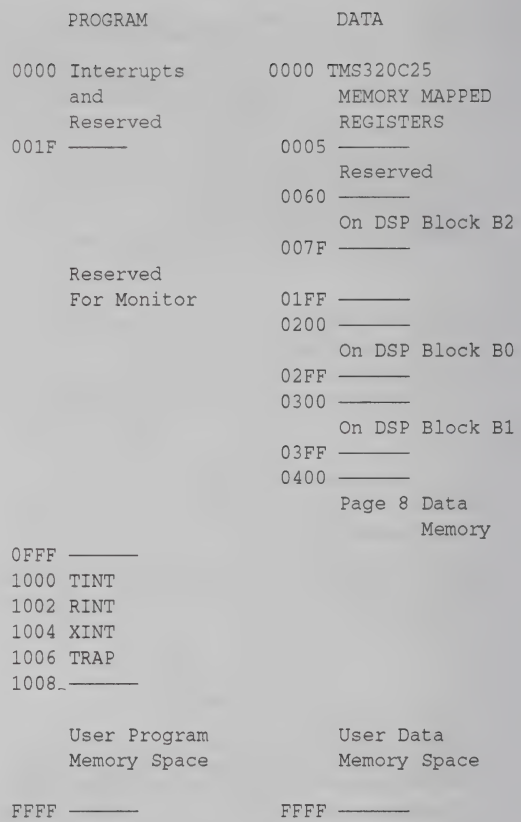
The next basic step is to get a good feel for the basic architecture of the DSP-93. Spend some time examing the schematics and get a good look at the interconnect lines. The basic system includes a DSP engine board and a radio/computer interface board. The DSP Engine, contains the TMS320C25 DSP, 32K by 16 bits of program and data memory - upgradable to 64K, the clock circuitry (40MHz) and some programmable array logic for system I/O. The Radio/Computer Interface Board, top board, contains two eight pin female mini-DIN connectors for radio interfacing. Incoming radio signals pass through a voltage divider to establish the initial levels, then through an eight channel multiplex chip. The multiplex chip then feeds the single A/D input with either of the radio inputs or one of the six auxiliary inputs. The Texas Instruments TLC32044 Analog I/O chip is used to sample and update the input signal at a rate of up to 45K operations per second and includes aliasing filters. This board also communicates with your computer at speeds up to 19.2K baud using a serial connection and, with special programming, this can be increased to the maximum rate attainable by a 16C550 and your computer. [2,3]

The next step is to begin to get comfortable with either of the development systems (Mac [5] or Windows [4]) and play with assembling available source. A low cost shareware assembler, TASM TMS320-25 Assembler, is available for code development and both the Mac and Windows interfaces provide near seamless use of the assembler. Code development and testing can become a quick task.

What follows is a basic overview of various areas needed, when developing for the DSP-93.

Memory Map

The following memory map exists after you have entered a 'G' command from the Monitor. The 'G' command is executed after a program download using DSPLOAD.EXE or the windows program D93WE. The unit is operating entirely in static RAM and in the high speed mode at this point.



The Harvard architecture used in the TMS320 is quite different from the typical microprocessor. There are two memories utilized during one instruction—data and program. Until experience is gained working with this architecture, it is easy to forget this basic principle. There are two memories active at the same time. By having two memories, a single instruction can load a word out of program memory and do something with a word out of data memory, all within the same cycle. It facilitates implementing filters and other digital signal processing algorithms. For example, stepping down a table of constants, such as filter coefficients and doing a multiply and add to accumulator with a data array of signal values can be accomplished at full machine cycle speed with no overhead for instruction fetches, nor double accessing of a single memory.

Program and data memory can change, and keeping track of "what is which" is needed. How this is done is largely a function of how the DSP-93 design uses the TMS320. Therefore, the TMS literature will not give the whole story necessary to understand the DSP-93.

The DSP-93 has a number of physical memories. Some memory is internal to the TMS320C25 chip and other is external. The tables below outline the physical memories, as well as the conditions which determine whether they are used by the TMS320C25 as program or data.

Internal memory (TMS320C25)

Label	Can be configured as	Size
B0	Data or program *	0100h
B1	Data only	0100h
B2	Data only	0010h

* B0 is in data mode after a hardware reset, Monitor reset, or CNFD instruction. It is program memory after a Monitor 'G' command, or CNFP instruction.

External memory

How the external memories are configured depends on the state of the XF bit. Hardware reset turns the XF bit ON, as will the SXF instruction. RXF turns it OFF. SRAM U104, U105, U110, and U111 come with the basic kit and provide two 32K memories (one program and one data). Four more SRAM IC's will raise the amount to 64K for both program and data memory.

(1) - XF line/bit is ON after a hardware reset, after a Monitor reset command ('R'), or after a SXF instruction. RXF turns the bit off, and it is also turned off after a Monitor 'G' command.

(2) - Data addresses below 0400h access internal memory in the TMS320 and therefore are not available for use in the external memory.

(3) - Addresses FF00h - FFFFh access internal memory, B0, when B0 has been configured as program memory (normally after a Monitor 'G' command).

Note that all memory is organized as 16 bit words, and not 8 bit bytes. Along the same lines remember that the accumulator is 32 bits long so be careful about unintended sign extension.

Internal memory, B0, is configured to the data mode after a hard reset. Also, the Monitor in EPROM configures B0 to data when a 'R' (reset) command is executed, and to program when a 'G' command is executed. Switching B0 modes is done with the instructions CNFP and CNFD, (set program, and set data, respectively), or the hardware reset that switches it to data.

External memory is switched via the XF line out of the TMS320. This line is controlled by the XF bit. A hardware reset sets this bit high. The bit can be set/reset by the instructions SXF/RXF, respectively.

IC number	XF = ON(1) "Slow speed"		XF = OFF(0) "Fast speed"	
SRAM				
U104, U105	Data	0000 - 7FFF(2)	Prog	0000 - 7FFF(2)
U106, U107	Data	8000 - FFFF	Prog	8000 - FFFF(3)
U108, U109	N.A.	.... -	Data	0000 - 7FFF(2)
U110, U111	N.A.	.... -	Data	8000 - FFFF
EPROM				
U102, U103	Prog	0000 - 7FFF(2)	N.A.	.... - ....

When the XF bit is high, such as after power-up reset, the EPROM, which contains the Monitor programs, is active as program memory. This makes it possible to get the machine running with something intelligent. XF also selects a low speed mode, so that the processor is slow enough to accommodate the EPROM. XF also makes external memories U104,5,6,7 switch to data mode and U108,9,10,11 not accessible. This arrangement allows the EPROM Monitor to load program into data memory (U104,5,6,7 and B0). This switching is necessary since the TMS320 does not execute instructions which store anything into program memory; the downloaded program is placed into the DSP-93 as data.

With the Monitor program executing out of EPROM, the usual step is to download a program from a general purpose computer. The program being downloaded is stored as data, in data memory, which is U104,5,6,7 at this time. Upon completion of the downloading, the general purpose computer issues a 'G' command which causes the DSP-93 Monitor to turn off the XF bit. This switches U104,5,6,7 from data memory to program memory. The Monitor also jumps to location 1008h (of program memory) to start the program that was just loaded (into what was data memory).

With XF low, program normally runs out of external SRAM, U104,5,6,7, using U108,9,10,11 for data storage. B0, B1, and B2 can also be used. B0 as stated before can be configured either as data or program, though remember that the Monitor sets it to program after the loading process completes and a 'G' command is given. Operating out of internal memory is somewhat faster than external memory and may be needed for time critical operations. The TMS320C2x User's Guide shows timings for instructions according to the memory combination being used.

Since B0 can be switched between data and program, it can be loaded with a program which can be executed. During the normal program downloading process, B0 is configured as data, so it can be loaded with program no differently than U104,5,6,7. When the loading process completes, the Monitor does a 'G' command and jumps to 1008h. The program at 1008h can then jump to B0 and execute the code which was loaded. When B0 is configured as program, it is no longer at locations 0200h - 02FFh, but occupies FF00 - FFFFh. Therefore, the code assembled at 0200h - 02FFh must be capable of executing properly when moved to locations FF00h - FFFFh. The jump from the program in U104,5,6,7 will be to FFxyh, if the beginning of the code loaded into B0 is 02xyh.

External memory is not accessed for data addresses below 0400h, as these are reserved for the TMS320. Also, FF00h - FFFFh of program memory is not accessed when B0 is configured as program.

Note that some addresses are really registers within the TMS320, such as locations 0 and 1 which are used to load/receive the serial shift register data to/from the AIO chip. The TMS manual covers these in detail.

External memory, (prog, fast), 0400h - 1000h holds the Monitor which is used when in the fast mode. If these locations are blasted, such as with a Monitor Fill command, the Monitor is lost and a hard reset is required.

The TMS320 cannot load and store (i.e. move) data from program memory to program memory. Data-to-data memory can be accomplished, as well as program-to-data and data-to-program. Therefore, it is not likely that a runaway program will blast the Monitor stored in 0400h - 1000h.

The Monitor uses B0 during the transition from EPROM to the SRAM or from low speed operation to high speed. Therefore, it is not possible to load the entire 256 words with program, or Fill it via the Monitor. If a reset command is executed by the Monitor, locations 0200h - 0275h are overwritten with "stuff" from the Monitor, wrecking what might have been downloaded into B0. As long as a Monitor reset does not occur between the loading of B0 and utilization of B0 (either as data or program), then the full page can be used. Otherwise, only those locations not used by the Monitor can be used (0276h - 02FFh).

DSP-93 Firmware - Monitor Operation

When the DSP-93 is powered up, the firmware Monitor takes control of the unit. The Monitor conditions the TMS320C25 and then begins polling the serial data link looking for single character instructions to execute. The action taken by the Monitor during each operation will be explained in more detail here.

After initialization is complete, the Monitor enters a polling loop checking the serial port for an input character. The Monitor is not case sensitive and an entry of '?' will bring a listing of the Monitor version and the commands. The commands available in the monitor are as follows:

A-AR REGISTERS
This command displays the value contained in the eight 16 bit registers on the TMS320C25. The first register is used by the Monitor and so the value in register zero will normally be 0400 hex. This is what you should see when you enter 'A' after power up.

D-DUMP MEMORY
The dump memory command displays the specified block of program or data memory. You must specify whether you want to use program or data memory. Then, enter the hex starting address and the ending address of the memory block. The memory contents will then be presented in block form with one 16 bit data address and eight 16 bit data values on each line.

F-FILL MEMORY
The Fill memory command works like the data dump command except it is filling memory with a 16 bit hex value. If you write over any program areas, you will kill the Monitor.

G-FLIP & RUN PROGRAM @ 1008h
The 'G' command is used after you have loaded a program and you are ready to run it. The Flip referred to here means that the Monitor is flipping from the EPROM over into the static RAM.

H-INTEL LOADER HIGH BITS
This is an Intel style hex loader for placing bits in data memory. To execute properly, the DSP-93 must be in the LOW speed state. The data is expected in the following sequence; count (8 bits), byte address (16 bits) (the byte address is twice the word address), Intel hex code command (8 bits), some quantity (count) of data (8 bits) and finally an 8 bit checksum. If no checksum errors are found, the data is placed into memory in the address locations specified. If an error occurs, a checksum error message is transmitted via the serial port. When the loader is finished, control is returned to the Monitor.

J-JUMP TO XXXX & RUN
JUMP and run executes in the same manner as the 'G' command except execution begins at the specified address. The interrupt vectors must still be in place if you are going to allow interrupts to occur.

L-INTEL LOADER LOW BITS
The 'L' command works exactly like the 'H' command except it deals with the lower 8 bits of the 16 bit words.

M-MODIFY WORD
This command is used to change the contents of a 16 bit memory location in program or data memory. The command can be used to force the D/A output of the TLC3204X to a particular value. This can be done by modifying location 0001. D/A changes will only occur if the AIO chip is active.

P-PROGRAM
This command is used to launch one of the firmware programs located in the DSP-93 EPROM. '?' will list all programs available.

R-RESET
Entering an 'R' will cause the DSP-93 to go through a soft reset. The results of this should be equivalent to a hardware reset. If you have entered the 'G' command and you are working in static RAM, the DSP-93 will bounce back to the EPROMs just as if you hit the reset button. Some of the areas in data RAM are initialized during a reset cycle.

S-SHOW WORD
This command is used to display the contents of a particular memory location.

T-MEMORY TEST
This command tests the current data memory for errors. The memory is tested by writing 0000h, 5555h, and 0FFFFh into every location and reading it back. The test will loop through all locations and give a '*' prompt if no errors occur. If an error occurs, the location of the error will be reported along with the value written and the value returned. Both RAM banks can be tested by issuing the 'T' command in slow mode (i.e., after a reset) and again in fast mode (i.e., after a 'G' command).

Sine/Cosine table
In versions of the Monitor prior to Version 2.17, the sine/cosine table located in EPROM was copied to RAM memory when the 'G' command was issued. Beginning with Version 2.17, although the table is still in EPROM, it is no longer copied to RAM. This 1) frees up some Monitor code area 2) eliminates the possibility the sin table will write over code the user loads, and 3) allows user to locate the sin table in data memory which is the place he will really need it.

The table in EPROM is a 540 degree, 0.25 degree step, sine/cosine table is located in EPROM from 767Fh to 7EEFh and is scaled by 2^15. Sine begins at 767Fh; cosine begins at 77E7h.

The Parts of a DSP-93 Program

One of the best ways to learn DSP-93 programming is to read, study, and understand existing programs. A lot of source code is provided on the system diskettes. Use this valuable resource. Not every program will use all these parts, and not all that do use them will use them exactly as shown. If they did, there would be only one program!

Header (Instructions, Copyrights, Disclaimers)
It's a good idea to have a header that tells what the program is, how to assemble and execute it, who wrote it and a Copyright and Disclaimer statement. See the sample program for more examples.

The Include Files
There are currently five include files that define most of the constants that you will use when programming the DSP-93. It is *strongly* encouraged that all programs include these files and use the constants therein. This will make your program easier to understand and will reduce programming errors. These files are:

MACROS.INC
Purpose: This macro defines the origin in the code segment that the initialized data is to begin. This macro defines storage in the code segment while also creating a symbol representing the location the data will reside in data memory after an initialization block move.

MONITOR.INC
Purpose: This file defines the DSP-93 Monitor functions and addresses. Symbolic names for all of the entry points have been assigned. This file should be used instead of the absolute addresses since a linker does not exist. The file may also contain any macros defined in the future that can help in using the Monitor functions.

PORTS.INC
Purpose: This file defines the DSP-93 I/O ports and I/O bits. The file assigns symbolic names for each of the ports and bits. The user should make use of the TASM bitwise AND (&) and OR (|) operators to manipulate the bits. The programmer is discouraged from using hex values in programs, as other users may have to read the listings, and the symbolic names assign greater meaning to the code.

SERIAL.INC
Purpose: Define symbols used in configuring the serial port for the DSP-93. The symbols in this file describe the 16550 UART. Programmers should use this file instead of using magic numbers in their programs.

REGS.INC
Purpose: This file defines the register replacements for the TASM assembler. The register file AR0 through AR7 are defined as well as the memory mapped registers defined, by the processor.

The following lines of code should be in your source after the header:

```
          .NOLIST
#include  "MACROS.INC"
#include  "REGS.INC"
#include  "PORTS.INC"
#include  "MONITOR.INC"
#include  "SERIAL.INC"
          .LIST
```

Memory Location Equates

Storage variable locations in internal and/or external memory must be defined in your program. These variables are defined using an .EQU directive to assign values to labels. For example:

```
;  global variables
BUFI   .EQU   060h   ;AIO input buffer
BUFO   .EQU   061h   ;AIO output buffer
DO     .EQU   062h   ;data output buffer
```

Program Constant Equates

Constants used in your program should be assigned a label and that label be given a value using an .EQU directive. It is *strongly* encouraged that all programs use labels for constants rather than using constants in the body of the program.

Program Origin

The TASM assembler does not, by default, set the DSP-93 starting address of 1000h. So be sure to include the directive: `.ORG 1000h`

Interrupt Vectors

The first four instructions of the program handle the various interrupt vectors and must branch to the appropriate labels. The Timer and Trap interrupts will not occur in the DSP-93 programs, so they branch to the program starting label GO. Program execution begins at 1008h, just following the TRAP interrupt vector.

```
                ;Define Vectors
     B   GO     ;branch to program
                ;  start (TINT) Timer
     B   RINT   ;AIO receive interrupt
                ;  service routine
     B   XINT   ;AIO transmit interrupt
                ;  service routine
     B   GO     ;branch to program
                ; start (TRAP)
GO   DINT       ;program starts here
```

Initialization of the Math, Serial and Memory Model

There are various parameters for specifying how mathematical operations, the DSP chip serial IO, and memory models will be handled. Your program should set these values at the beginning of the program.

Initialization of the AIO

The initialization of the AIO chip is probably the most confusing aspect of DSP-93 programming. However, using the recipes written by the development group should enable you to start the chip sampling at the rate you desire without problem.

The AIO chip is reset and enabled by manipulating the IO data lines D15 and D14. In doing so, the variable CFG in which the value to be output to the RADIO_GAIN port must be in external memory, i.e. page 8 or greater.

Immediately after resetting and enabling the AIO chip, it must be configured to specify the AIO gain, sync, filters and the values of RA/TA and RB/TB which set the sample conversion frequency. See "AIO Port Programming, Setting the AIO conversion frequency" [1].

Initialization of the DSP

The initialization of the DSP consists of initializing of any program values your program may use. For example:

```
DSP_INI   ZAC          ;  zero A

          SACL   DO    ;  zero buffer
          SACL   SO    ;  zero buffer
```

Handling Interrupts

The transmit and receive interrupts, generated by the AIO chip, must be handled by your code. This requires two functions RINT and XINT, pointed to by the interrupt vectors at the beginning of your program. Data received from the AIO are stored in the variable BUFI and processed, in this example, by the function DSP. Data to be sent to the AIO is stored in the variable BUFO and will be processed when a transmit interrupt occurs. TINT and TRAP interrupts can be handled in the same way.

Serial Port IO and Exiting to the Monitor
If your program is to read the DSP-93's Serial Port while executing, code is available that provides that capability. The function should be called from someplace in your code that is executed repeatedly. The character read is returned in the variable CHARREAD. If no character were available, ACC will be zero upon return. In any case, to be a user-friendly DSP-93 program, include this function always. If an upper- or lower-case R is sent to the DSP-93 Serial Port, the program will exit to the Monitor.

Wait Functions
There are two "wait" routines in the Monitor (See MONITOR.INC). There are also three "wait" functions that are commonly included in DSP-93 programs. These have delays of:

WAIT4 104 msec
WAIT2 52 msec
WAIT1 26 msec

Defining Data Tables
Tables of data such as filter coefficients, strings, etc. may be defined within your program.

Using Pre-defined Data Tables
There are two tables of waveforms provided with the DSP-93 source code. One table is the Sin/Cos table:

```
0..511 is Sin(theta), 512 words = 2 PI
64..639 is Cos(theta), 512 words = 2 PI

512 words long for each table, total
640 words long.  The table is scaled by
2^11
```

the other is the general waveform table:

```
Table 0 = Sine(theta) scaled by 2^15
Table 1 = Triangle(theta)
Table 2 = Square(theta)
Table 3 = Sawtooth(theta)
```

The End
Don't forget the following directive at the end of your program.

```
          .END
; end of program
```

AIO Port Programming

Setting the AIO conversion frequency
The AIO conversion frequency (sampling frequency) is set during the AIO configuration. There are two values that determine the conversion frequency, TA and TB. The conversion frequency (in Hertz) is:

$$\frac{10{,}000{,}000}{2 \cdot TA \cdot TB}$$

The values are most easily set using the macros CMDA_VAL and CMDB_VAL.

See "Linear Circuits; Data Conversion, DSP Analog Interface, and Video Interface; Data Book Volume 2" for more information on setting the conversion frequency. The AIO chip has a specified upper limit on the conversion frequency of 19.2 kHz, but the chip will operate considerably in excess of this. For example, the 9600 bps FSK modems use a conversion frequency 41666 Hz. A table of conversion frequencies is provided in the programming guide [1].

IO Port Programming

Which Port does What
The TMS320C25 has 16 IO ports, many of which are implemented in the DSP-93. Data is written to an I/O port with the OUT instruction and read with the IN instruction. For example:

```
OUT     CHARREAD,UART_CTRL
IN      CHARREAD,UART_READ
```

Debugging DSP-93 Programs

Debug functions in the Monitor
The Monitor ROM includes a debug routine developed by Tom McDermott, N5EG. This routine can be called as a development aid when you are generating new DSP code or if you are just studying existing code.

Debugging using LOC.ASM
The code in LOC.ASM was developed for finding problems. It may be used to determine when program execution stops. An author of code which seems to be stopping mysteriously should

integrate this code into his program for testing. Then one of the users with a DSP-93 which hangs may run the modified program and report back with the results. Results can be reported by dumping memory in the DSP-93 using the Monitor. The memory dump can be captured and posted for evaluation by the author of the code. If we can determine when the problem is occurring the solution may also appear.

Monitor IO Routines

A collection of Monitor routines which should help speed code development is available for use.

Refer to the file MONITOR.INC included with the release disks for the exact location of these routines.

GET4HEX and GETVAL16
These two routines are the same. They collect a 16 bit hex value from the serial port. The value should be presented as four ASCII characters (0 through F digits). The four hex digits are converted to a sixteen bit HEX word and stored in a single memory location for later use. Look at MONITOR.INC for the storage location and a suggested label. This memory storage location is for the current page of memory established by the LDPK instruction.. The LDPK is normally set to page 8 with the instruction LDPK 8.

GET2HEX
This routine collects an 8 bit hex value from the serial port. The value should be presented as two ASCII characters (0 through F digits). The two hex digits are converted to an eight bit HEX word and stored in a single memory location for later use. Look at MONITOR.INC for the storage location and a suggested label. This memory storage location is for the current page of memory established by the LDPK instruction.. The LDPK is normally set to page 8 with the instruction LDPK 8.

GETCHAR
This routine collects a 4 bit hex value from the serial port. The value should be presented as one ASCII character (0 through F digits). The hex digit is converted to a four bit HEX word and stored in a single memory location for later use. Look at MONITOR.INC for the storage location and a suggested label. This memory storage location is for the current page of memory established by the LDPK instruction.. The LDPK is normally set to page 8 with the instruction LDPK 8.

HEXOUT
This routine assumes a single HEX digit is stored in the accumulator. This digit is converted to ASCII and sent to the serial port.

INBIT
This routinue retrieves incoming data from the serial port. Data is placed in the indirect address location pointed at by AR(ARP).

OUTBIT
This sends the lower 8 bits of AR(ARP) to the serial port. This routine uses memory location 0060h of the current page for temporary storage. Location 0060h is used to store the UART register information while the routine is sending the data. The register information from the UART is used to determine if CTS is set or cleared and if the transmit data buffer is full.

OUTBIT2
Sends the lower 8 bits of location 0062h of the current page to the serial port. This routine uses memory location 0060h of the current page for temporary storage. Location 0060h is used to store the UART register information while the routine is sending the data. The register information from the UART is used to determine if CTS is set or cleared and if the transmit data buffer is full.

PRVAL08
The 8 bits in a defined memory location of the current page are converted to ASCII and sent to the serial port. MSB is sent first. See MONITOR.INC for the proper memory location and label to use for this routine. This routine uses HEXOUT.

PRVAL16
The 16 bits in a defined memory location of the current page are converted to ASCII and sent to the serial port. MSB is sent first. See MONITOR.INC for the proper memory location and label to use for this routine. This routine uses HEXOUT.

RESET
Jump to this location to restart the Monitor.

SD_CRLF
Sends a single carriage return and line feed to the serial port.

SD_STR
Calling this routine with a 16 bit address stored in the current AR(ARP) register will cause a string to be sent to the serial port. The C-style string should begin at the AR(ARP) memory pointer and end with 00. Only the lower 8 bits of the memory words will be sent. The string data must be in data memory space.

SP1 through SP10
These routines will send the indicated number of spaces (20 Hex) to the serial port.

MWAIT1 and MWAIT_A
The WAIT routines are shown here. You can enter the routine at MWAIT1 or MWAIT_A with your own value set for AR2. With a 40 MHz clock, the wait from MWAIT1 will be about 32 milliseconds while the wait from MWAIT_A will be about N mS where N is the value in AR2. Use MONITOR.INC for the entry locations for MWAIT1 or MWAIT_A.

MP_BLK_MV
This routine will move a block of data memory to program memory. You must provide pointers to the beginning and end of the block in program memory space and the beginning of the block in data memory space. This routine would be useful if you are dynamically creating program code in data memory and need to move it to program space.

PM_BLK_MV
This routine will move a block of program memory to data memory. You must provide pointers to the beginning and end of the block in program memory space and the beginning of the new block in data memory space. This routine would be useful if you want to include a sin table in your program. Your program would be loaded into the DSP-93 in what will become program memory. After starting your program you would use this routine to move the table to data memory. The routine may also be used to move string data from program memory to data memory for later use. The following example shows how the routine would be called.

Assembling a DSP-93 Program

How to assemble a file using TASM under DOS
If you are using DOS, the command line to assemble a TASM source file, say XXX.ASM, is: `TASM -3225 -la1 -g0 XXX.ASM`
This will result in a listing file XXX.LST being generated and an object file XXX.OBJ. For further options when using TASM, see the TASM documentation.

Windows and Macintosh Assembly
See the Help files for information on using D93WE or DSP-93Control as a code development environment.

References:

1. Parsons, Ron, W5RKN. (1995). Programming Guide for the DSP-93. June, 1995. TAPR: Tucson, AZ.

2. Stricklin, Bob. (1994). TAPR/AMSAT Joint DSP Project: DSP-93. Proceedings of the TAPR 1994 Annual Meeting. Tucson Amateur Packet Radio Corp.

3. DSP-93: The TAPR/AMSAT Joint DSP Program. (1994). Issue #55, Summer 1994, Packet Status Register. pp. 1-4. Tucson Amateur Packet Radio Corp.

4. McDermott, Tom, N5EG. (1995). D93WE Windows Development Environment for the TAPR/AMSAT DSP-93. Proceedings of the 1995 TAPR Annual Meeting. St Louis, MO. March, 1995. TAPR: Tucson, AZ. p. 31.

5. Parsons, Ron, W5RKN. (1995). DSP-93 Control Macintosh Development Environment for the TAPR/AMSAT DSP-93. Proceedings of the 1995 TAPR Annual Meeting. St Louis, MO. March, 1995. TAPR: Tucson, AZ. p. 38.

An Introduction to FlexNet

Gunter Jost, DK7WJ
FlexNet Gruppe Darmstadt
Lichtenbergstrasse 77, D-64289 Darmstadt, Germany

Donald Rotolo, N2IRZ
Radio Amateur Telecommunications Society
PO Box 93, Park Ridge NJ 07656, USA

Abstract

Features and operation of FlexNet packet networking software are discussed. Details of the software architecture, RMNC and MS-DOS hardware platforms, applications, user interface, adaptive parameters, and routing techniques are presented.

Introduction

FlexNet is a flexible, modular and user-friendly software program for packet radio networking. First conceived in 1987, the software has undergone many changes and improvements since then, and is presently (July 1995) at version 3.3. The most notable features of FlexNet are:

- Autorouter: Automatically routes all connect requests with minimal input required of the user.
- Adaptive Parameters: All network parameters (except TXDelay) are automatically set by the software, and adapt to changing channel conditions.
- Hop-to-hop Acknowledge: Each packet is ack'd directly by its neighbor, improving data transport reliability.
- Command Interpreter: The software interacts directly with the user, and has numerous internal applications providing detailed data concerning the configuration and status of the node.
- Remote Controllability: The node is completely remote-controllable, also allowing for remote tracing of a QSO with several filters.
- Modular and Portable Architecture: over 95% is written in C, allowing wide portability across platforms.
- DAMA master and slave implemented. All channels may be master or slave, or mixed as desired.

FlexNet is presently the most widely used packet radio networking software in Germany, with over 500 installations. Due to German amateur radio regulations, each unattended station (node) must receive special permission to operate, and must use tightly coordinated dedicated point-to-point links. Despite creating problems and red tape, these limitations have forced the creation of a very high-quality network, where round-trip response times over hundreds of kilometers and tens of node sites are typically only a few seconds.

Although FlexNet is primarily used for networking, the PC version can also be used by users by simply eliminating the Node module. PC/FlexNet is thus a flexible yet powerful replacement for permanent TNC emulations (such as TFPCX). It is extremely simple to set up, since there are no hassles with parameters.

Hardware

FlexNet is presently ported to two hardware platforms: The Rhein-Main Net Controller (RMNC) system and the Intel 80x86 processor running MS-DOS.

The RMNC platform is a 6809-based system using a Z8530 SCC. Each channel control card (one for each radio/data channel) is on a standard Eurocard, and is plugged into a standard card cage with backplane bus. One master card can control up to 15 slave cards. Developed by the Frankfurt (Rhein-Main) Packet Radio Group, the RMNC remains the preferred platform for FlexNet because of its low cost and high performance. The software resides on a single EPROM.

An MS-DOS version has existed since 1990, but was not distributed, as it was used primarily as a test and development platform. In 1994, development of a stable version for MS-DOS was begun in earnest. FlexNet runs problem-free in an XT-class system, although faster processors are preferred. Several different channel drivers exist, allowing for some flexibility in I/O hardware.

The choice of MS-DOS was based primarily upon the large installed base. Admittedly, a better choice would be LinuX, and in fact the task of porting FlexNet over is already being worked upon.

Architecture

In 1994, a discussion between the present FlexNet author (DK7WJ) and BayCom author Flori Radlherr DL8MBT yielded a new concept for FlexNet: Modularity. The major portions of the software were developed as separate modules, the most important being the channel drivers and applications. These modules can be used in combinations as desired, on either hardware platform, creating a very flexible system, and allowing a larger number of authors to contribute talent. In the future, a "developer's kit" will be available, further easing module development.

A number of driver modules presently exist:

- USCC — Driver for all BayCom SCC cards. The card type is automatically detected.
- SER12 — Driver for the serial BayCom AFSK modem.
- KISS — Driver for connection to a computer. This driver should not be used with a TNC, as the channel timing is not as precise as it should be, especially in the half-duplex mode. FIFO UARTs are supported, allowing data rates up to 115,200 baud on a 286/16.
- 6PACK — Driver for TNCs. 6PACK is a new protocol that overcomes timing problems with KISS and TNCs. This protocol can control up to 8 TNCs in a ring (without tokens!), however the data rate on the RS-232 side should be greater than the sum of the RF baud rates. Beta-testing is currently underway, involves replacing the TNC's EPROM.
- PAR96 — Driver for the parallel version of the BayCom 9600 baud G3RUH-compatible modem
- IPXN — Driver for Novell-IPX networks. Those that already have such a network running can link PCs using FlexNet. Traffic runs in the local net segment as broadcasts, and each FlexNet station monitors the net traffic.
- IPXPD — The same as IPXN, except that the packet driver is replaced with the IPX protocol itself. To be compatible with an IPXN system on Ethernet, only used when a Novell installation is not available.
- IPPD — AX/IP, a provision of IP to allow point-to-point connections over Ethernet. ARP is rudimentarily implemented. Not a solution for an internet link, but functional. Eventually , this will be integrated into a TCP/IP module.
- VANESSA — Driver for the VANESSA channel card. This card from the Swiss SEPRAN group can control 2 RF channels and has its own processor, significantly reducing the PC's workload.

Other drivers being developed include those for TMC320C25 DSP systems as well as Sound Blaster cards. Interested software authors are invited to contact DK7WJ for further information about developing additional drivers.

Applications

The FlexNet system supports several applications, primarily to provide users and Sysops with information about the network or a node. These applications are mentioned in "User Interface" below. Due to the modular design of FlexNet, implementing new applications is relatively easy, and can be written by anyone having sufficient programming experience.

Three unique applications deserve special mention:

- TFEMU — Hostmode emulation. PC/FlexNet can be used with this application as a driver for various hostmode programs, such as BBS, DXCluster, terminal programs, etc.
- ETHEREMU — FlexNet can be used with this application to emulate an ethernet packet driver, allowing for example NCSA-Telnet or Winsocket applications to be worked via AX.25 Datagrams or Virtual Circuits.
- RDOS — Remote DOS interface for service. This application allows full remote control of MS-DOS, even such local utilities such as formatting a disk.

User Interface

FlexNet appears as a simple interactive application to the user. Various commands are used to set up connections, access applications, etc. The user connects to the node's user port and sends the desired command.

To initiate a connection, the user need merely specify the destination, although other options including manual routing, single-stepping from node to node and partial autorouting are possible.

The commands available to the user are summarized below. A complete description of all commands is given in the Sysop documentation, presently available in German and English.

<A>ktuell (Current_info)
Displays a current-interest text uploaded by the sysop.

<B>eacon
Displays the Beacon text. This file shows which beacon is sent at what interval from which ports.

<C>onverse
Starts FlexNet Converse mode (if sent without a callsign following, otherwise see CONNECT below). 255 channels are available. Function and usage of the FlexNet Converse mode is similar to other popular conference systems.

<C>onnect
Starts a connect request to another station when followed by a callsign. A connect command can also be established by the user without being connected to the node, by specifying the entry and exit points of the network, and the destination. The node routes (see "Routing" below) SABM frames toward the destination, and sends the message "link setup..." to the user. If the connection is ack'd by the destination station, the user receives the message "*** connected to <call>". When unsuccessful, the message "*** failure with <call>" is sent to the user. If a DM is received from the destination, the message "*** busy from <call>" is sent to the user. The connect request can be broken by sending a <CR> or a new connect request. If a user attempts to connect to a station he is already connected to, he receives the message "*** can't connect twice". This command can also be used to connect to a different user port at a node site that has more than one user port. For example, the command "C-7" connects the user to the port with SSID 7, which is acknowledged with the message "*** <call>: ssid ok"
After using the Connect mode, if the other station disconnects, the user is reconnected to the last node connected to, which is announced with the message "*** reconnected to <call>".
It is not permitted to manually route a connect request to form a loop, where the connection passes through the same node twice. If an attempt is made, the user is informed with "*** <call> loop detected", and reconnected to the previous node.

<D>estinations
Shows the automatically-generated destination table. Information includes the callsign, SSID range, measured ping time (100ms steps) and, optionally, the entire route to the destination.
It should be noted that the destination list is nearly identical in every node in the network, since all updates are propagated throughout the network immediately.
The user can also selectively view the destination table, for example "D HB9" shows all destinations having callsigns beginning with HB9.

<F>ind
This command, one of the most powerful in FlexNet, sends a request to all nodes in the network (can be limited by the sysop, see also Setsearch) to find a particular station. Each node sends an UNproto frame, addressed to the station to be found. If the station hears this frame, it will send a DM frame. When this occurs, a message is sent back to the user exactly where the desired station can be found. If there is no response, the user receives no message. Since UNproto frames can get lost, the user should send the command a few times to be sure. Naturally, the AX.25 protocol requires that the correct SSID be used.

<H>elp
Sends a help file uploaded by the sysop.

<I>nfo
Sends an information file uploaded by the sysop

<L>inks
Lists the sysop-defined link table, showing all ports, neighboring nodes, and their status.

<LO>cal
Sends the node's connect text file. Always sent when connecting directly to the user port, this command allows the text to be viewed remotely.

<M>heard
Displays up to 200 heard callsigns with time stamp and channel number.

<MY>call
Displays the node's callsign and SSID range.

<P>arameters
Displays a detailed list of all ports at the node, their parameters, various link statistics and their status.

<Q>uit
Disconnects the user from the node, after the node sends "73!".

<S>etsearch
Shows which nodes will transmit the Find message.

<U>sers
Lists all users of the node, including those just passing through. Shows QSO number (internally generated), connection status, data frames unack'd, port, and callsigns involved.

<IO> (Input/Ouptut)
Shows status of each of the 16 binary inputs and outputs (RMNC version).

Sysop Commands:
The following additional or expanded commands are available exclusively to the sysop:

<L>inks	Set links list
<M>ycall	Set callsign and SSID range
<P>arms	Set various parameters
<IO>	Read all input lines and/or set all output lines
<CA>	Send a CALibrate signal
<K>ill	Kill a specific QSO
<MO>de	Set HDLC parameters

<SY>sop	Sysop authentication
<W>rite	Upload various text files
<T>race	Monitor all activity on a channel remotely
<RESET>	Cold reset
<RESTART>	Warm reset

Connection Phases

There are four possible phases for any given connection or QSO: Link Setup; Information Transfer; Link Failure; and Disconnect. Each phase is discussed briefly:

1. Link Setup

During a connection setup, the connect (SABM) frames are simply routed through the network (see "Routing Methods", below), without being directly acknowledged (No UA Frame). Thus, a connection can only be made when the destination station is actually available, for it is the only station that will acknowledge the SABM frame. Of course, the user can connect to any node instead of a user station, and then attempt to manually route the remainder of the path from there. However, the fully automatic routing method is significantly easier for automated stations, such as BBS store-and-forward operations.

When the called station acknowledges the connection with a UA frame, it is passed back (again without any hop-by-hop ack) to the calling station. At this point, there are two QSOs in the QSO list on every node involved: one from calling station to destination station, and one in the reverse direction. Once the calling station receives the destination station's UA frame, the link is established (as a Virtual Circuit) and information transfer can begin.

The reason for this method, instead of the more conventional direct acknowledge for SABM frames, is that very few network resources are wasted on connect requests that cannot be established, yet good connections are established fairly quickly. In the FlexNet network installed in Europe, typical response times from end to end are on the order of a few seconds, so timeouts, lost packets and other problems are rare.

2. Information transfer

During the information transfer phase of a QSO a 'Hop-by-hop Acknowledge' is used. Each I-Frame (packet frame containing some information) from one or the other station is directly acknowledged by the node, and then passed along to the next node along the path (also directly ack'd). Retries caused by whatever reason on a given link are thus only retries across one link and not across the entire path.

Once the path is setup, it is completely transparent to the data flowing through it. Each frame passes through the network completely unchanged. Thus, data frames with other Protocol IDs (PIDs), such as TCP/IP, are carried across the network without any problems.

3. Link failure

If, during information transfer, the path is broken for some reason, the end nodes report the link failure to both stations with the message "*** DB0xxx --> Link Failure" and the session is disconnected.

4. Disconnect

If station A sends a disconnect frame, it is immediately ack'd by the local node. The node then attempts to sever the connection to station B. If there is no data remaining 'in the pipeline' for station B, the disconnect occurs immediately. If data remains, it is first passed to station B and then the link is disconnected. If station B had sent data to station A, it is lost.

Routing Methods

To connect it is only necessary for the user to specify the point of entry to the network and the destination, although any connection may be manually routed if desired. There are four methods available to automatically route the users connect request:

1. Routing by destination table
2. Routing by link table
3. Routing by Heard list
4. Routing by SSID

Routing is only performed for the Connect (SABM) frame. Once the Virtual Circuit is set up (the UA of the SABM is received by the calling station), all subsequent frames take the exact same path. If a failure occurs, the link is torn down after informing both ends of the failure. A failed link must be reestablished manually.

1. Routing by destination table

The first method is based upon routing by callsign information. The node looks at the destination callsign and compares it with a table of calls and routes that has been stored in a destination table by the autorouter. If a match is found, the call is passed along to the specified neighbor.

[NOTE: All node callsigns begin with DB0...]

Example: DB0ODW has the following links:
1: DB0KT
2: DB0AAC
3: DB0IE
and knows the following destinations: DB0EQ, DB0ZDF, etc.

The frame
<fm DG3FBL to DK7WJ via DB0ODW,DB0ZDF>
is expanded to:
<fm DG3FBL to DK7WJ via DB0ODW,DB0AAC,DB0ZDF>

This is because DB0ODW knows that DB0AAC (which it has a direct link to, on Port 2) is the next (and best choice) node along the path to DB0ZDF. Thus, the frame is sent via Port 2, despite the fact that there is no direct link to DB0ZDF. The 'best' path is always selected for any given destination, as determined by the average 'ping' time for a given path.

The node sends out test frames ("Pings") to each neighbor, and then stores the round-trip response time. Using the ping information, the destination tables are constantly updated in real time. If a destination becomes available (or unavailable), that information is instantly broadcast throughout the network (limited only by the propagation time through the network), thus all destination tables are nearly identical. The destination table is built automatically and cannot be changed by the operators. It is thus assured that only accessible destinations are forwarded, and instantly deleted when they (or their path) disappear.

2. Routing by link table

If the autorouter does not find an entry in the destination table, it then goes to the sysop-defined link table. If a pre-defined route is found here, the frame is sent to the specified neighbor (on a specified port) for handling. The link table normally contains only the direct neighbors, that is, those nodes having a direct link with the node.

Example: DB0ODW has the following links:
1: DB0KT
2: DB0AAC
3: DB0IE
4: DB0AIS

The frame
<fm DG3FBL to DK7WJ via DB0ODW,DB0AIS>
would be routed to Port 4 so long as no entry exists in the destination table, because DB0AIS is known in the link table. Hopefully, DB0AIS would have a route to DB0ODW.

3. Routing by Heard list
The node stores the last 200 directly heard callsigns in an internal list. This list remains in memory until a RESET occurs. An entry to this list occurs only when a station has an active connection, or when it is found using the 'Seek' command. At first, the node uses the SSID of the desired station to establish a connection but, if there is no response, then the SSID is ignored. It is therefore without problems possible to be in STBY on various ports with various SSIDs, so that one can always be found on the correct port. Incoming connect requests or also UNproto frames are routed with this list.

4. Routing by SSID
A final chance for the node to automatically route frames is based upon the SSID. Specific channels can be arranged by SSID (this can also be done by Port number). This is shown with the PARMS command. To use the SSID routing, the user specifies the SSID (Port) that the frame should be sent from.

Example: DB0ODW has the following arrangement of SSID versus Channel numbers: Channel 1 has SSID 0, Channel 5 has SSID 12, Channel 6 has SSID 3, etc. If a connect frame arrives that has specified an SSID for DB0ODW, the frame is routed directly to the channel specified by the SSID:

The frame <fm DG3FBL to DK7WJ via DB0ODW-12> is routed to channel 5, which has the SSID of 12, so long as no other way to DK7WJ is known. The following frame is sent out from channel 5: <fm DG3FBL to DK7WJ v DB0ODW-3*>, which clearly shown where the frame came from (in this case, channel 6 has the SSID 3). This also carries the entry node information. This turnaround of SSIDs is also notable, in that every frame shows where it came from.

One of the most important features of FlexNet is that all connections are reversible, that is, you are always provided sufficient info to have the same connection in the reverse direction.
This routing method is naturally used above all for user access.

If all of these routing methods fail, the connect frame is lost. If this occurs when the user is connected to the node, the message "*** <callsign>: can't route" is sent to the user. If the user issued a connect command when disconnected from the node, the TNC will eventually reach the RETry limit.

Adaptive Parameters

All level 1 and level 2 parameters within the network (including user access channels) are either self-adjusting according to the current channel conditions or fixed in the software. The only exception to this is TXDelay, which is set by the sysop.

A RETry limit of 10 is used, to ensure a quick recovery for links subjected to temporary interference. Regular polling is used to verify that links are operational, as well as to determine the actual one-way or round trip data transfer time. The polling interval, controlled by FRAck (T1), varies according to channel conditions. If FRAck is small, polling occurs no sooner than every 90 seconds, lengthening when FRAck becomes larger. Each node polls only its neighbors. The round-trip turnaround time is used to adjust FRAck (T1) to the time it takes for the neighboring station to acknowledge I-frames, and is also used (along with other factors) by the autorouter to determine the best (fastest) path to other stations.

MAXframe is regulated according to the capability of the link. When the other station receives all frames without problems, MAXframe runs as high as 7. At 1k2 baud, 7 frames in a row are seldom sent, since the maximum key-down time is limited to about 12 seconds. When multiple streams are active, the 12 seconds is divided up and shared. When the other station begins missing frames (for example by sending reject frames), MAXframe is reduced. In this case, the throughput is actually increased with a lower MAXframe. It should be noted that evaluation checks the link quality TO the other station, not FROM the other station, and that the value changes as necessary. A poll, in which all frames are acknowledged, or with lost acknowledgements, don't reduce MAXframe.

P-Persistence is the critical parameter when many users are on the channel simultaneously. A fixed parameter is, at best, a compromise between collision potential and transmit potential. FlexNet notes the number of users on the channel, as well as the data rate and other factors, and adjusts P-Persistence to offer the best

performance at all times. Aggressive stations no longer have the advantage of faster performance at the expense of weaker stations, yet fast downloads remain feasible when channel activity is low. The improvement over a fixed parameter is most notable on duplex user ports (the most common in Germany), but improvement is noticeable on simplex channels as well.

Concerning TXDelay, if the node hears more than 100ms of flags from a user station, which indicates an excessive TXDelay setting in the users TNC, a polite message is sent informing the user of this fact, and then disconnects. Thus, a user running excessive TXDelay cannot use the network until the problem is corrected.

It would be highly desirable if user software were to also follow the trend towards full adaptation to the channel conditions. It is clear that a computer can adjust to prevailing conditions faster and more accurately than a human, not to mention the fact than most users are totally baffled by TNC parameters anyway. Other problems also are caused by using a single fixed parameter to control a constantly changing channel, which would be resolved with adaptive user software.

PC/FlexNet can be installed as user software, to take full advantage of adaptive parameter adjustment, by simply omitting the Node module. It becomes a powerful yet flexible replacement for permanent TNC emulations like TFPCX - simply use FlexNet with TFEMU and channel drivers as needed. Setup is simple, with only one parameter (TXDelay) requiring operator input.

Interoperability with TheNET

It is relatively easy and painless to include TheNET network nodes into a FlexNet network. Each TheNET node that interfaces directly with a FlexNet node appears in the FlexNet destination table. The interface must be single-stepped manually, but no serious problems occur as TheNET users are used to single-stepping ;-)

For Further Information

This overview is necessarily brief. Complete details are available with the software (as an ASCII file) or in separate printed documentation available from the author. At this time (July 1995) the documentation is available in German as well as English.

References:

G. Jost, DK7WJ and J. Sonnabend, DG3FBL, *"FlexNet, the European Solution"*, Proceedings of the 9th ARRL Computer Networking Conference, pp127-133, 1990.

G. Jost, DK7WJ, *"FlexNet Sysop Manual v3.3"*, 2nd edition, 1995.

Convolutional Decoders for Amateur Packet Radio
Phil Karn, KA9Q

Abstract

This paper describes two freely available convolutional decoders written in the C language. One implements the Fano algorithm for sequentially decoding three rate 1/2 K=32 codes while the other implements the Viterbi algorithm for maximum likelihood decoding of the rate 1/2 K=7 NASA Standard code. Both support 8-bit soft-decision decoding with arbitrary metric tables and perform according to theory.

Using GCC 2.6.3 under BSDI/OS 2.0 on an AMD 486DX4-100 CPU, the Fano decoder runs up to 375 kb/s while the Viterbi decoder runs at a constant speed of 75.1 kb/s. On a 90 MHz Intel Pentium, the speeds are 594kb/s (Fano) and 118 kb/s (Viterbi). This is fast enough to be useful in many amateur packet radio applications.

The code is available over the Internet. See this web page for details: http://www.qualcomm.com/people/pkarn/ham.html

Introduction

Convolutional coding is one of the two major types of error control coding (the other is *block coding*, e.g., Golay, Reed-Solomon). A convolutional encoder is very simple. It consists of a K-stage shift register into which user data is clocked. The value K is called the *constraint length* of the code. The shift register is tapped at various points according to the code *polynomials* chosen. Several tap sets are chosen according to the code *rate* given as a fraction, e.g., 1/2. The denominator of the fraction (2 in this case) is the number of code polynomials while the numerator (1) tells how many user data bits are shifted into the register per encoder cycle.

Each set of tapped data is summed modulo-2 (XORed) and becomes one of the encoded output symbols. For example, in the K=7 rate 1/2 code I use in my Viterbi decoder, user data is clocked into the 7-stage shift register one bit at a time. The first encoded symbol is then formed by modulo-2 summing the outputs of shift register stages 1, 3, 4, 6 and 7, while the second symbol is formed by modulo-2 summing stages 1, 2, 3, 4 and 7. Then the next data bit is clocked in and the cycle repeats.

Convolutional encoders are extremely simple; that's one reason they're so popular on deep space probes. Convolutional *decoding* takes more work. Basically, all convolutional decoders work by generating local hypotheses about the originally transmitted data, running those hypotheses through a local copy of the encoder, and comparing the encoded results with the (noisy) encoded data that is actually received. A decoder can pursue one hypothesis at a time as in the sequential decoder, or it can pursue many in parallel as in the Viterbi decoder. Both decoders compute *metrics* as estimates of the "closeness of fit" between the received and locally regenerated symbols, and they provide cumulative path metrics with the decoded packet as a useful estimate of the decoder's "confidence" in the result.

Unlike block codes, convolutional decoders can easily use *soft decision* information from the demodulator about the quality of each received symbol. They easily handle variable packet sizes such as those found in computer networks, and they're fairly easy implement in software (though they don't necessarily run fast!)

This is not to say that convolutional codes are always superior to block codes. Block codes have some complementary advantages, such as the ability to handle larger data blocks and a greater ability to handle burst errors. The two are sometimes *concatenated* to combine their strengths, as in the *Voyager* and *Galileo* spacecraft.

But one has to start somewhere, so I implemented some convolutional decoders to see if they could run fast enough on personal computers to be useful in amateur packet radio. I believe I have been successful.

Development Environment

Error control decoding can take a lot of CPU cycles. My Fano and Viterbi decoders are respectively 1 to 2 orders of magnitude slower than my DES encryption code, which is usually thought of as very compute-intensive. Careful optimization thus becomes extremely important to overall system performance.

I use BSDI/OS, a commercial port of Berkeley (BSD) UNIX to the 32-bit Intel processors (386/486/Pentium) and a close cousin of FreeBSD and NetBSD. BSDI supports a pure protected-mode 32-bit environment that is much more efficient for FEC than the 16-bit "real mode" provided for DOS backward compatibility.

The performance gains from a 32-bit environment are simply too great to pass up. Given that the 386 is already virtually obsolete and the 486 is now on its way out, this doesn't seem like an unreasonable assumption. My code does not assume BSDI; it should run well in other 32-bit environments, including 32-bit DOS extenders. Since it is in portable C, it should also execute in 16- and 8-bit environments, but I expect it would be rather slow.

The Fano Decoder

My first coding project was a sequential decoder using the Fano algorithm. Sequential decoders explore one hypothetical data sequence at a time by locally encoding it and comparing it with the noisy encoded version that is actually received. As long as they match reasonably well, the decoder moves forward. When the received and local versions differ by more than a certain amount, the decoder backs up, tries an alternative data hypothesis and moves forward again to see if it works better. The process repeats until the decoder reaches the end of the packet or exceeds some time limit.

I chose a code with a constraint length K of 32. Sequential decoders can easily handle such large constraint lengths, and 32 was a natural choice given the CPU word size. I originally chose the NASA Standard polynomials used by the Pioneer 10 and 11 probes. I then found two other polynomials in the literature with slightly better performance, so I added these as compile-time options.

Sequential decoders have been heavily analyzed, and their behavior is well understood. As might be expected, decoding time is a random variable that depends strongly on the quality of the received signal. When the Eb/N0 is several dB more than the decoder threshold (2.5 dB for this code), decoding quickly proceeds at a nearly constant rate. (See fig 1). Near the threshold, the mean decoding time and its variance increase. (See fig 2.) Below the threshold the variance becomes infinite and many packets "time out" the decoder. (See fig 3).

An important figure of merit for any Fano decoder is its "computational rate", the speed at which it can examine a branch in the code tree and make a decision to move forward, backward or sideways in the tree. On a "clean" packet the decoder rapidly moves forward, but a noisy packet causes many backward and sideways moves that slow down the process.

Optimizing the Fano decoder

Optimizing the Fano decoder involves a tradeoff between "clean" and noisy packets. I decided to optimize for noisy packets, as long as the resulting hit for clean packets wasn't too great. After all, clean packets already decode quickly. Noisy packets need the most help. Some optimizations required additional memory, but on modern systems these were no-brainers.

The main part of the Fano decoder is a loop that executes on every forward motion. Imbedded in this loop is a smaller loop that does backward motion when needed. Optimizing this code was fairly straightforward. It consisted mainly of "factoring out" operations from the loop whenever possible so they're executed only once per packet no matter how many moves the decoder makes. For example, all possible metrics for each branch (four for this code) are computed before going into the loop, even though only two are actually examined on any particular forward decoder motion. Although this represents a little extra work on a clean packet, this helps substantially on a noisy packet by avoiding the repeated recomputation of branch metrics.

Another optimization is to stash quite a bit of information into the data structure that represents a node in the decoding tree. Each node element includes the decoder state and cumulative metric along with the list of all possible branch metrics just mentioned. This lets the decoder move rapidly backward by simply decrementing a pointer, without having to recompute anything. This improves performance on noisy packets at the cost of extra memory and a slight reduction of decoding speed on clean packets.

The Viterbi Decoder

In contrast to the sequential decoder, a Viterbi decoder *logically* explores in parallel every possible user data sequence. It encodes and compares each one against the received sequence and picks the closest match. For this reason, it is also known as the *maximum likelihood* decoder.

The decoder can't actually do this, of course, since the number of possible data sequences doubles with each additional data bit. But the Viterbi decoder recognizes at each point that certain sequences cannot possibly belong to the maximum likelihood path, and it discards them. This happens because the encoder memory is limited to K bits; a Viterbi decoder in steady-state operation keeps only 2^(K-1) paths.

The complexity of a Viterbi decoder therefore increases exponentially with the constraint length K. Larger K's do perform better against noise, but a Viterbi decoder for a K=32 code is just not practical. For my Viterbi decoder I chose the rate 1/2 K=7 NASA standard convolutional code used on Voyager and in many commercial satellite systems.

This shorter code has less coding gain than the K=32 sequentially decoded code, but Viterbi decoding has the practical advantage of executing at a constant speed regardless of the received signal-to-noise ratio. This makes it attractive for real time, delay-limited applications that can tolerate some uncorrected errors, e.g., digital speech. Viterbi decoding is also more tolerant than sequential decoding to metric table and receiver AGC errors. And the inherent parallelism of the Viterbi decoder makes it easy to implement in hardware, an important consideration when contemplating very high speeds. It is not yet clear, though, that these advantages outweigh the slower average execution speed when implemented in software in a packet radio system. Experimentation with both is warranted.

The structure of the Viterbi decoder is very different from the Fano decoder. The Fano decoder consists of several pages of fairly "random" C code. But the Viterbi decoder consists almost entirely of a simple Add/Compare/Select (ACS) operation that is executed repeatedly for each decoded data bit -- 64 times per bit for a K=7 code.

As the name implies, an ACS operation adds a branch metric to each of two cumulative path metrics that converge into a node, compares them and selects the "surviving" (best) path. This implements the "discard paths that can't possibly be right" feature at the heart of Viterbi decoding.

By analogy, consider a highway routing problem. If you determine that the route from Baltimore to New York via Philadelphia is shorter (better) than the route that goes through Pittsburgh, then you don't really have to consider the Pittsburgh route at all when extending your trip beyond New York to Boston. The same thing happens in distance-vector routing algorithms such as RIP and NET/ROM: each node only propagates its best routes to its neighbors.

An important design parameter for a Viterbi decoder is the path memory length. An ideal decoder would keep every possible path in memory and delay a final decision about the first bits of the packet until the very end. But this isn't really necessary. Analysis and simulation have shown that several constraint lengths back, the paths usually merge into a single maximum-likelihood candidate. So a Viterbi decoder that retains 4-5 constraint lengths of decoded data will achieve nearly the same uncorrected error rate as an ideal decoder.

For a K=7 code, a 32-bit path memory is a nice match to a 32-bit word size. The ACS operation can record the surviving path at each node by simply copying the single machine word containing it. This is fast and efficient since 32-bit operations on a 32-bit machine are no more costly than smaller operations.

Optimizing the Viterbi Decoder

ACS operations are often paired into "butterflies", so called because of their appearance on a trellis diagram. There are 32 butterflies per data bit for a K=7 code. As you might guess, the butterfly macro was the focus of almost all of our optimization efforts.

Optimization focused on two goals: minimizing the number of CPU cycles per butterfly and maximizing the L1 (on-chip) cache hit ratio. This latter goal is especially important on the 486DX2 and DX4 since external bus cycles are proportionately even more expensive in a clock-multiplied CPU.

Originally I executed the butterflies in a tight loop, 32 loops per data bit, giving the expected poor performance. The most dramatic speedup came from unrolling this loop, but not for the reason you might suspect. The usual reason to unroll a loop is to amortize the loop control overhead over more instructions. This contributed a little, but the big win came from something else.

The ACS operations work on an array of node elements. There are two such arrays, one for the decoder's current state and another for the new state after the current symbols have been processed. After each received symbol pair is processed, the two arrays are swapped. (Actually, pointers to these arrays are swapped). Every 8 bits, the path with the best metric is chosen

and the high order 8 bits are extracted and placed in the output buffer.

Each butterfly works on a fixed set of old and new state array elements and a fixed pair of hypothesized branch symbols. The original code computed the addresses of each array element and the branch symbols as functions of the loop index, which meant doing it repeatedly at execution time. But when the loop was unrolled, constants replaced the loop index in these expressions. This allowed them to be completely evaluated at compile time and replaced with immediate constants in the generated code, giving the dramatic speedup.

Around this time I upgraded my CPU from the clock-doubled 486DX2-66 to a clock-tripled 486DX4-100. (The latter really should have been named the 486DX3-99...) On many CPU-intensive operations this new chip gives nearly the 50% improvement you'd expect from the ratio of internal clock speeds. This was true for my DES code and for the Fano decoder, but the Viterbi decoder only sped up by about 15%.

A little analysis revealed why. The 486's on-chip cache operates in write-through mode. For programs that only occasionally write to memory, this isn't a problem because the CPU also has a 4-deep memory write queue. As long as there's room, the CPU can schedule a write and continue execution. But if the queue is full, the CPU blocks until space opens up.

Each Viterbi ACS reads five memory words: two node metrics, two branch metrics and the surviving path. It also writes two words to the new node: the survivor's path and metric. This is an unusually high ratio of memory writes to memory reads. The reads generally hit in the cache, but the writes go through to main memory. If there's a bottleneck in writing to main memory, this would explain why the DX4-100 isn't much faster than the DX2-66: the external memory bus is no faster for the faster CPU.

Unfortunately there is no easy way to disable write-through caching on the 486. Fortunately, it's easy to do on the Pentium; most systems apparently support write-back caching. This is a big win in Viterbi decoding because the data written to cache in one cycle is immediately read in the next and overwritten in the one beyond that. There's no reason to go to main memory at all.

Not having a Pentium, I stopped further optimization efforts. Franklin Antonio N6NKF, who does have a Pentium, picked up the code and tried a few more tricks. He examined the assembler code generated by his compiler (Watcom) and discovered that by reordering the butterflies to place together those operating on the same pair of branch symbols, the compiler did a better job of allocating registers and keeping common subexpression results involving the branch metrics. This produced a noticeable speedup that to my surprise carried over somewhat to the 486.

Acknowledgements

I would like to thank Franklin Antonio, N6NKF, for the considerable time he spent optimizing the Viterbi decoder code on his Pentium, and for pointing out some bugs.

Appendix - sample decoder performance

This section presents performance results for the two decoders. In the Fano section I present information about the decoder's variable speed while in the Viterbi section I show the uncorrected error performance. (Since Fano decoders on long constraint length codes produce essentially no errors and Viterbi decoders execute at a constant rate, there's little point in the reverse set of statistics.)

These results assume nonfading additive white gaussian noise (AWGN) channels and ideal BPSK or QPSK modems. I used the classic "rejection method" (see *Numerical Recipes in C*) to produce normally distributed (gaussian) noise samples from a conventional uniform random number generator.

Keep in mind that without coding, ideal BPSK and QPSK both require an Eb/N0 of about 9.6dB for a bit error rate of 1e-5.

Sample Fano decoder performance

Statistics from several runs of the Fano decoder follow. To test the effects of metric table inaccuracies, the driver program allows for different values of Eb/N0 to be used in the computation of the metric table than for the actual generation of the noisy test signal. Here the same values are used for each. The threshold adjustment parameter *delta* is an internal tuning

parameter of the decoder that depends on the scaling of the metric tables; *delta*=17 was found to work well by experiment.

The value *N* represents the number of forward decoder motions per data bit required to decode a particular packet; for each run, both an average and a cumulative histogram are given. For example, in the Eb/N0 = 3dB run the average number of forward decoder motions per user data bit was 2.46, while 2.2% of the packets took 6 or more motions per bit. To estimate the throughput of the decoder at a given Eb/N0, simply divide the rate for "clean" packets (375 Kb/s on the 486DX4-100) by the average N for that Eb/N0. This ignores the cost of backward motions, but they're relatively cheap.

The decoder imposes a limit on *N* for each packet; if this limit (10000 for these runs) is exceeded, the packet is erased (discarded). In Fig 3 note that even though 33% of the packets were erased, none were decoded in error. This is a nice side benefit of long constraint length codes: separate error correction mechanisms like CRCs are not really required.

```
Figure 1 - Fano decoder performance 2.5 dB above threshold

Seed 806575651 Amplitude 100 units, Eb/N0 = 5 dB metric table Eb/N0 = 5 dB
Frame length = 1152 bits, delta = 17, cycle limit = 10000, #frames = 1000
decoding errors: 0
Average N: 1.181747
  N >=  count fraction
     1   1000 1
     2      0 0
```

```
Figure 2 - Fano decoder performance 0.5 dB above threshold

Seed 806575707 Amplitude 100 units, Eb/N0 = 3 dB metric table Eb/N0 = 3 dB
Frame length = 1152 bits, delta = 17, cycle limit = 10000, #frames = 1000
decoding errors: 0
Average N: 2.464074
  N >=  count fraction
     1   1000 1
     2    540 0.54
     4     71 0.071
     6     22 0.022
     8      9 0.009
    10      6 0.006
    20      3 0.003
    40      0 0
```

```
Figure 3 - Fano decoder performance 1.5 dB below threshold

Seed 806575769 Amplitude 100 units, Eb/N0 = 1 dB metric table Eb/N0 = 1 dB
Frame length = 1152 bits, delta = 17, cycle limit = 10000, #frames = 1000
decoding errors: 0
Average N: 561.711378
  N >=  count fraction
     1   1000 1
     2   1000 1
     4   1000 1
     6   1000 1
     8   1000 1
    10    997 1
    20    977 0.98
    40    929 0.93
    60    890 0.89
    80    859 0.86
   100    837 0.84
   200    762 0.76
   400    677 0.68
   600    620 0.62
   800    576 0.58
  1000    556 0.56
```

```
  2000     489 0.49
  4000     413 0.41
  6000     378 0.38
  8000     351 0.35
 10000     326 0.33
```

Sample Viterbi decoder performance

This section shows the results from a typical run of the Viterbi decoder. The Eb/N0 here is 3.5 dB; this value was chosen more to produce a reasonable amount of output than anything else. The encoded data was all zeroes, so errors show up as "1" bits in the decoded data. (Decoded data is not shown for the 97 frames that decoded correctly). Note the characteristic bursts in which the errors occur, even though the channel is AWGN and stationary. Burst errors are well suited to Reed-Solomon block codes, which is why Reed-Solomon and Viterbi-decoded convolutional codes are sometimes concatenated.

```
decoded data:
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000033000000000000000
00000000000000000000000000000000
decoded data:
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
002a240000000000000000000000000
decoded data:
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000010000000000000000000
Seed 806709882 Amplitude 100 units, Eb/N0 = 3.5 dB
Amplitude 100 units, Eb/N0 = 3.5 dB
Frame length = 1152 bits, #frames = 100, bits decoded = 115200
frame errors: 3 (0.03)
bit errors: 10 (8.68056e-05)
```

Data Radio Standard Test Methods

Burton Lang, VE2BMQ
Donald Rotolo, N2IRZ

North East Digital Association
PO Box 563, Manchester NH 03105

Abstract

The Data Radio Standard Test Methods document is introduced and explained. The document consists of a number of standardized test methods, written in a clear, step-by-step format. Each test method is designed to be easy to perform, yet yield meaningful results. The rationale and organization of the DRSTM are discussed, including the proposed DRSTM database of measurement data.

Introduction

With the ever-increasing demands of modern packet networks upon the radios being used, there exists no standardized group of test procedures for these radios, or even a list of which radio characteristics are important, and why. If we are to maximize the efficiency of the shared radio networking environment we use, then we must have a better understanding of the significant characteristics of the radios being used. In an effort to resolve this apparent lack of information, VE2BMQ began writing the Data Radio Standard Test Methods in August 1994. In October 1994, the Board of the North East Digital Association endorsed this effort as beneficial to advancing the radio art, and encouraged other organizations to do so as well. In November 1994, the Board of the Radio Amateur Telecommunications Society also endorsed the DRSTM. In this paper, these efforts will be summarized.

Why do we need this?

Recently, a test report of radios advertised as "9600 Baud ready" appeared in QST [May 1995, pp 24-29]. The author, Jon Bloom KE3Z, provided a detailed analysis of the bandwidth considerations for proper operation at 9600 baud, also briefly mentioning deviation and its effects. He then provided test results of the Bit Error Rate (BER) for a few radios, a critical yardstick of data radio performance. However, the BER is measured in a continuous data stream, without the normally found pauses between groups of packets. These pauses, which are set by TXDelay (transmit delay) can significantly affect the overall performance of a packet radio link.

As an example, compare two commonly used 70 cm radios: The TEKK T-Net Mini (Model KS-900L) and the ICOM IC-449A. The TEKK has a RX/TX turnaround time of less than 12 milliseconds, while the ICOM has been measured at over 410 milliseconds. At 9600 baud, this translates to a loss of data capacity on the order of 80%. From the table below, we can see that a 2400 baud link with a TXDelay of 40 mS has a higher throughput capacity than a 56 kb link with a TXDelay of 500 mS!

Baud Rate	Byte Time	Throughput (bps) given a TXDelay of: 0mS	40mS	250mS	350mS	500mS
1200	6.67mS	127	121	99	91	81
2400	3.33mS	254	**233**	163	143	121
4800	1.67mS	506	431	241	199	158
9600	.833	1015	750	316	248	187
56k	.104	8131	2124	435	316	**223**

[Reprinted from NEDA 1994 Annual, pg.81. Assumptions: 230 Byte data per 256 byte transmission, 16 byte acknowledgement, both transmitters have same TXD]

The point is that one parameter that was not even measured in the test has a huge influence upon the performance of a data radio in a real packet network. If such a serious omission can be found in a technically competent journal such as QST, it can be said that most amateurs have an incomplete understanding of the parameters affecting data radio performance.

Organization

The Data Radio Standard Test Methods (DRSTM) is organized much like the published procedures of large standards organizations, such as the ASTM, IEEE or SAE. Each test method first defines the parameter to be measured, then explains its importance to data transmission. A detailed, step-by-step test procedure is then provided, along with set-up diagrams and a standardized form for recording and interpreting the test results. This ensures that all test results, whatever their source, are as reliable as possible. Efforts are taken to use commonly available test equipment wherever possible. This allows as many people as possible to perform the tests. The tests themselves are kept as simple as possible, while still yielding meaningful results, which allows a wide range of radio equipment to be tested. While we believe that many radio users would not actually perform measurements, just knowing how a particular parameter is measured can offer insight to its effects.

In addition, the DRSTM manual contains a detailed Glossary, a thorough explanation of the database structure, and performance requirements for all test equipment.

The scope and significance of each test that has been written so far is summarized below:

DRSTM-01 Transmitter Power-on Time Delay

Scope: This test procedure is intended to measure the time from the start of the transmitter keying (Push-To-Talk or PTT) line becoming active until the RF output has risen to 90% of its final value.

Significance: The time delay that a radio transmitter's power output has when it is keyed can range from microseconds to hundreds of milliseconds. A transmitter that is keyed but not putting out RF power can create a serious 'hidden transmitter' problem. In addition, in the case of a synthesized

transmitter, a significant difference between the Transmitter Power-on Time Delay and the Transmitter Power On-frequency Time Delay (see DRSTM-02) would indicate the possibility of serious interference to users on other nearby frequencies.

DRSTM-02 Transmitter Power On-frequency Time Delay, Frequency Stability Time Delay, and Modulation Stability Time Delay

Scope: This test procedure will measure three Data Radio characteristics:
1. Time from PTT becoming active to the RF output appearing within the designated passband of a test receiver.
2. Time delay until the transmitter output frequency has stabilized to within 5% of the channel bandwidth relative to its final stable frequency, or until the PLL loop tone or any other extraneous signal has decayed to less than 10% of the normal system modulation level, whichever is longer.
3. The time delay until the transmitter's modulation envelope has reached 90% of its final stable value.

Significance: The time delay until a transmitter's output frequency appears within its intended channel can range between microseconds and hundreds of milliseconds. Any difference between the Power Output Time Delay (see DRSTM-01) and the Power Output On-frequency Time Delay would indicate the possibility of serious interference to users on nearby frequencies. Any substantial delay in the appearance of the transmitter output signal in the receivers of other users on the same channel can result in a serious 'hidden transmitter' problem.

The additional delay of waiting for the frequency to settle down, or signals that have a significant amount of non-data modulation (such as the damped oscillation of a PLL oscillator feedback loop immediately following lockup) can have a serious effect on decoding the data signal. This could require a much longer TXDelay setting for usable operation.

Similarly, the time delay for the modulation envelope to settle down to its final stable value can also affect proper data recovery at the beginning of a data transmission. This rare condition has been observed in certain radios such as the Motorola MOCOM 35, where a large RC time constant in the reactance modulator retards the modulation envelope by several hundred milliseconds.

DRSTM-03 Transmitter Short-term Frequency Stability

Scope: This test procedure is intended to measure the transmitter and receiver frequency changes that occur when the radio is exposed to extreme temperature and supply voltage conditions.

Significance: Excessive changes of either the transmitter or receiver channel frequency can cause failure or degradation of packet radio links, if the change forces the signal beyond the passband of the received at either end. This information is important to designers of packet network facilities where equipment may be housed in unheated enclosures, as well as builders of emergency response networks that must not fail under extreme conditions.

It must be recognized that the quartz crystal used as the primary channel element or as a reference for a PLL is a major factor in frequency stability. Amateur radio users often use crystals from suppliers that may not follow the manufacturer's specifications. testing radios using such crystals can lead to more appropriate selection of crystals for particular radios.

DRSTM-04 Receiver Detector/Frequency/Squelch Recovery Time Delay

Scope: This test procedure measures:
1. The time between when PTT is released until a signal is received at the detector output
2. The time between when PTT is released until the receiver center frequency is within 5% of its design bandwidth relative to its final stable receive frequency, or until the PLL loop tone or other extraneous signal has decayed to within 10% of the normal demodulated data signal amplitude, whichever is longer
3. The time between when PTT is released until the receiver squelch circuit (and all preceding circuits) begin to pass a demodulated signal.

Significance: A useful data signal cannot be recovered from a received signal until the detector and all RF/IF/LO circuits ahead of it have recovered following a period of transmission.

Radios which take a long time for their frequency to stabilize, or that have considerable 'non-data' signals (such as the damped oscillation of a PLL oscillator feedback loop immediately following lockup) can have a serious effect on decoding the data signal. This could require a much longer TXDelay setting for usable operation.

Radio squelch circuits which take a long time to open following a transmit period would present decoding difficulties when the TNC or modem is connected after the squelch circuit (such as at the speaker jack). This would also require a longer setting for TXDelay.

DRSTM-05 Receiver Squelch Turn-on Time Delay

Scope: This test procedure measures the time between the start of a test signal until the squelch circuit in the radio opens and passes demodulated audio.

Significance: When using a TNC or modem connected after the squelch circuit, the time it takes for the squelch circuit to open and pass audio affects operation. If there is a significant time delay in opening the squelch, the TNC could decide to transmit before it realized the channel was occupied. This would cause a 'hidden transmitter' like problem.

DRSTM-08 Receiver Output Level, Impedance, Demodulated Frequency Slope (De-emphasis) and Demodulation 6dB Cutoff Bandwidth.

Scope: This test procedure measures:
1. The voltage level of the demodulated data signal output, per unit modulation level.
2. The impedance at the point where the demodulated audio output signal is connected.
3. The amount of audio output signal variation with frequency (De-emphasis slope) measured where the output is connected, expressed as dB per octave.
4. The audio frequency at which the demodulated signal drops to one-half the voltage on both the higher and lower sides of the normal operating frequency range.

Significance: The level and impedance of the demodulated audio signal from a data radio receiver is useful information when designing data systems and interfacing TNCs and modems. The de-emphasis response of a receiver affects TNC or modem operation. Most TNCs and modems require a flat frequency response while most radios offer some pre- and de-emphasis to improve voice quality. For optimum performance, these responses should be matched. The high side frequency response helps determine the maximum usable bit rate, while the low side frequency response is important with certain direct FM modulation systems (such as the G3RUH modem).

DRSTM-09 **Transmitter Modulation Drive Requirements, Input Impedance, Impedance Response Slope, Pre-emphasis Slope, Maximum Deviation and Modulation Frequency Capability.**

Scope: This test procedure measures:
1. The voltage level of the modulation signal per unit of transmitter deviation.
2. The input impedance of the modulation circuit.
3. The response characteristics of the modulator input impedance, as it varies with audio frequency.
4. The variation in the deviation when the modulating frequency is changed (pre-emphasis).
5. The maximum deviation capability without distortion.
6. The highest modulating frequency which does not reduce the first modulation sideband by more than 20dB.

Significance: The level and impedance of the modulator is useful when designing data systems and interfacing TNCs and modems. The modulator frequency response characteristic should be matched with the receiver response to obtain an overall 'flat' system response. A non-flat system response will distort data signals. The impedance variation of the modulator with frequency can also affect the modulator's frequency response characteristic. The maximum modulation (deviation) capability without distortion is useful when designing data systems and interfacing TNCs and modems. The maximum modulating frequency is useful in determining the maximum capabilities of a higher-speed data link, as well as in avoiding the radiation of wide sidebands produced by computer clock noise, etc. FM transmitters with direct connections to their modulators have been found radiating 10 MHz wide sidebands, caused by leakage of the TNCs 5 MHz CPU clock.

Database

An integral part of the DRSTM concept is a database containing the measurement results obtained by the DRSTM users. Each DRSTM enables the user to make the same measurements, consistently, and provides a form on which to record all measurements. It is well known that radios have differing characteristics for many parameters, even radios of the exact same type and of consecutive serial numbers. It is anticipated that, by providing an international clearinghouse and database of all measurements, that the amateur community would be better served. this may also provide an incentive for radio manufacturers to either publish the data-relevant parameters, or at least design radios with data transmission in mind. While no firm plans have been developed, it is expected that the database would be available on-line in some fashion, with free access for all.

Conclusion

The establishment of standards for various radio characteristics having significance in data transmission will eliminate much of the confusion and misinformation in the area today. These standard test methods, used to measure the performance of data radios, could be used by anyone having reasonable experience with common electronic test equipment. The international database would disseminate this collected data and, as manufacturers noticed that certain radios were unsuitable for data use, convince radio designers to modify their designs to acommodate data transmission. In this manner, the authors hope to improve the radio art.

It is the author's hope that other knowledgeable persons would step forward and offer their expertise in either writing standard methods, suggesting new tests, evaluating existing tests, or performing tests and disseminating their findings. At this time, these standard tests should be considered tentative. If you can help in any way, please contact the authors.

Modeling some data communications functions using Microsoft Excel 5.0.

Thomas C. McDermott, N5EG
Texas Packet Radio Society

Abstract

Recent enhancements to the Microsoft Excel[1] spreadsheet program, version 5.0, provide some interesting features that may be of interest to those designing or analyzing data modems. This paper looks at the following examples: 1) bit error rate of a modem vs. Eb/No in additive white gaussian noise (AWGN), 2) phase-locked loop response vs. loop filter parameters, and 3) modem eye patterns vs. channel response, and shows how each can be modeled with Excel.

Bit error rate in AWGN

To determine the theoretical bit error rate of a modem utilizing a certain modulation, it is necessary to know how often the noise voltage will exceed the signal voltage, given that the signal level (Eb), and the noise spectral density (No) is known. A common way to analyze this is to plot the bit error rate versus the Eb/No. In order to do this, an assumption about the statistical noise properties must be made. One assumption that is easy to model is that of additive white gaussian noise (AWGN) that has a uniform power spectral density (PSD) and a gaussian amplitude distribution. For an AWGN signal with zero-mean (no DC offset) and an R.M.S. voltage of 1 volt, the equation that expresses the probability density P(x) versus the voltage, x, is given by:

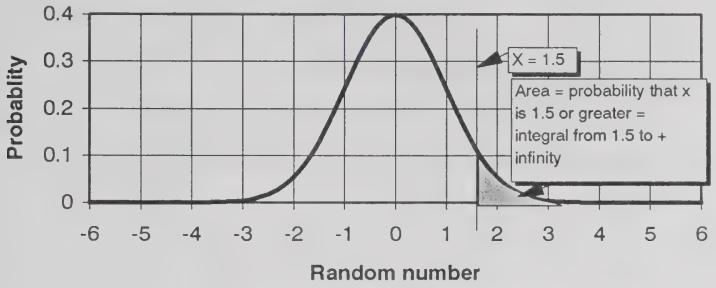
$$P(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{x^2}{2}\right] \qquad (1)$$

Then, the probability that the voltage x exceeds some value is the cumulative probability density of x versus the voltage. The probability that an error voltage will exceed the signal voltage is thus the cumulative density that the noise voltage is of the opposite magnitude and equal to or greater than the signal voltage. Assume that the signal voltage is n, then the probability of an error, versus the signal voltage, n is:

$$Error(x) = \int_{n}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{x^2}{2}\right] dx \qquad (2)$$

[1] Excel, Excel 5.0, and VBA are trademarks of Microsoft, Inc.

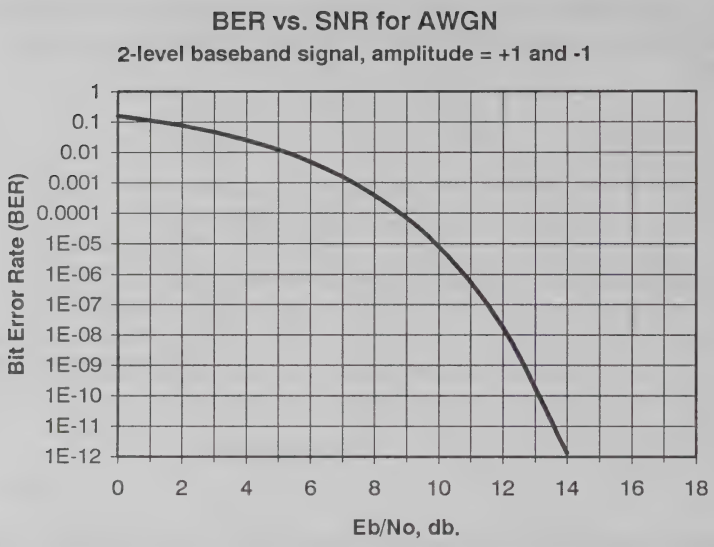
This equation is interpreted below, which shows the probability density of x (gaussian noise voltage) and the shaded part of the curve shows the cumulative probability of the noise from +1.5 to + infinity.



To plot the probability of the signal, n, being less than the noise voltage requires calculating the value of P(x) in equation 2 for all values of n. This is a tedious task, since equation 2 does not have an explicit solution. However, Excel 5,0 provides a function, called ERFC that provides the above probability. ERFC() is defined so that:

$$Error(x) = \frac{1}{2} ERFC\left(\frac{x}{\sqrt{2}}\right) \qquad (3)$$

Now, to plot the theoretical bit error rate (BER) of a signal vs. the Eb/No, all that is required is to plot Error(x) vs. x. The value x is generated as a ratio, and convenient decibel ratios are chosen for display (0 dB, 0.5 dB, 1 dB, etc.). This is plotted below, and is the theoretical BER vs. Eb/No (the value of x) for coherently-demodulated 2PSK modulation.



Different modulation formats can be compared by substituting into equation 3 the different formulas expressing x, the signal voltage.

Phase-locked loop (PLL) modeling

A simple phase locked loop consists of several components: a phase detector, a loop filter, and a voltage-controlled oscillator. In order to analyze the loop performance, it is necessary to express the phase detector gain, the loop filter gain, frequency, and phase response, and the VCO response. In general, the loop filter and the VCO response are complex (that is, they contain real and imaginary parts) and thus the equations must be computed using complex algebra. Two expressions of interest in the PLL design are the open-loop response, and the closed-loop response. The closed-loop response of a PLL is given by:

$$H_{closed}(s) = \frac{G(s)K_v K_f}{s + G(s)K_v K_f} \tag{4}$$

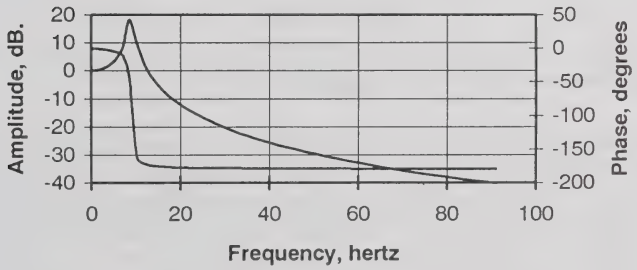
Where G(s) is the response of the loop filter, s is the Laplace variable (equal to $j\omega$ for sinusoidal analysis), Kv is the VCO control voltage gain, and Kf is the phase detector gain. Similarly, the open-loop response is given by:

$$H_{open}(s) = \frac{G(s)K_v K_f}{s} \tag{5}$$

To plot the magnitude and phase of H(s) versus the frequency, the use of complex algebra is required. Excel 5.0 supports complex numbers, and operators to add, subtract, multiply, divide, and to find the magnitude and phase of a complex number. These operations are not

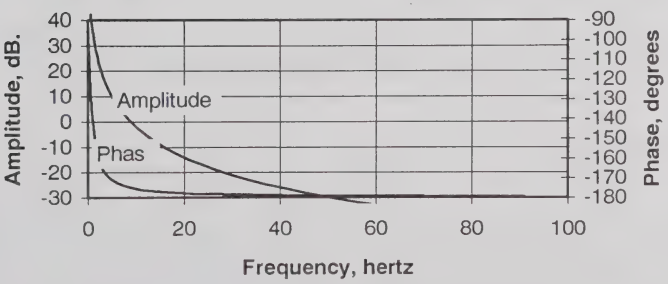
always as convenient as manipulating real numbers in Excel, so several steps are required in the computation. First, an array of numbers is set up as the frequency, f. Then the value of s is computed. s is equal to $j\omega$, or $j2\pi f$. j is equal to the square-root of negative one (an imaginary number). In Excel, s is equal to =COMPLEX(0,2*PI()*B9) where B9 happens to be the cell containing the frequency in radians per second (for this example). Thus, the real part is zero, and the imaginary part is 2*PI*f, and s obviously is j*2*PI*f. G(s), the loop-filter frequency response can be computed, and in the particular spread-sheet, each column is a different frequency, and each row is a partial product, such as s, G(s), and finally H(s). Then the magnitude and phase of H(s) are computed as additional rows, which can then be plotted versus frequency. The diagram below shows the closed-loop response of one such computation, a PLL with a simple low-pass loop filter. It can be seen that the loop response is very poorly damped, and the loop is near instability, with a gain peak near 9 hertz.

PLL closed loop response with single pole RC filter



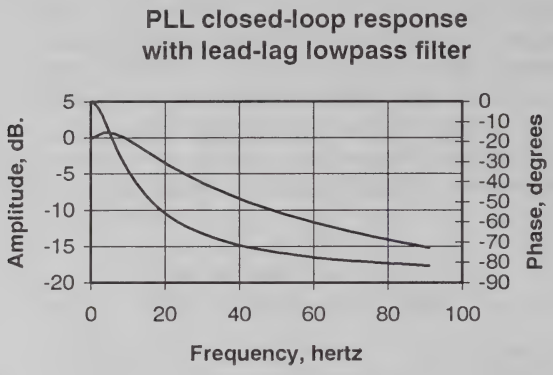
The open-loop response of this PLL is computed almost identically, and is shown below.

Open-loop response of PLL with simple low-pass filter



From the open-loop response it can be seen that the amplitude curve crosses zero-dB. at 9 hertz, and that the loop has about 6-degrees of phase margin, clearly an invitation to disaster with this loop. Fortunately, it is easy to make more complex lead-lag filters for the loop-filter,

G(s), and perform any additional calculations in Excel. It is also easy to make the pole frequencies of G(s) adjustable, so they can be altered, and the PLL response plotted accordingly. The diagram below shows the closed-loop response of a PLL with a lead-lag loop filter. This PLL is much more likely to operate properly.



Eye pattern versus modem channel response

A more complicated example, but one which shows the power of Excel is to compute the eye pattern that would be seen at a receiver given knowledge of the frequency and phase response of the channel. We'll assume a real frequency response, but this is not necessary (it does make this example less difficult). The key to this is provided by two features of Excel: the ability to compute the Inverse Fast Fourier Transform (IFFT), and the ability to write functions in Visual Basic Application (VBA) language that comes embedded within Excel 5.0. A function that is needed is the Multiply-Accumulate operation, which forms the kernel of convolution and correlation integrals.

Given the frequency and phase response of the channel, the impulse response of the channel is given by the Inverse Discrete Fourier Transform (IDFT) or the IFFT of the channel frequency response. If the frequency response has no phase variation, then the impulse response will contain only a real component. Once the impulse response is known, the time-domain response of the channel to a data signal can be computed by linear convolution of the data bits with the impulse response. The convolution function is given by:

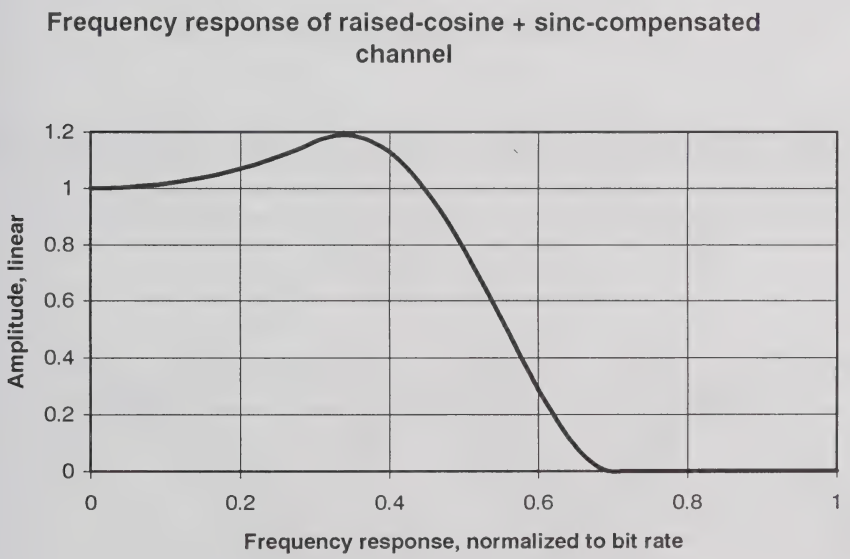
$$y(t) = \int_{-\infty}^{+\infty} h(\tau)x(t-\tau)d\tau \qquad (6)$$

Where y(t) is the output of the convolution, h(t) is the channel impulse response, and x(t) is the input signal to the channel. It can be recognized that when converted to discrete-time, this is just exactly the equation of a finite-impulse-response (FIR) digital filter (in fact, this is how the FIR filter is derived). The Multiply-Accumulate operation performs the multiplication of h and x for all values of tau, and sums them. This operation is then repeated for the next value of t. Thus, an array of Multiply-Accumulate (MAC) functions can perform a linear convolution (or

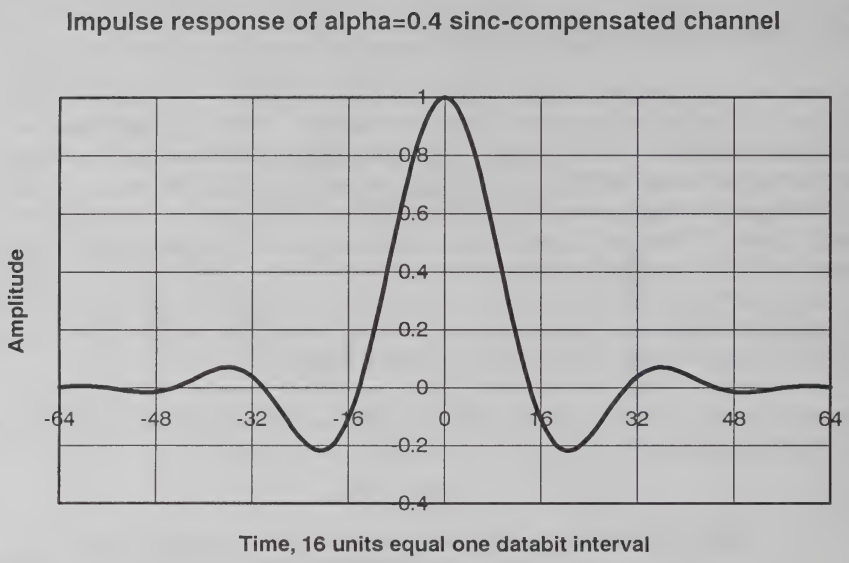
FIR filtering) of the signal. The MAC function written operates just like a built-in Excel function, and it can be copied and pasted in ranges.

To generate the eye pattern, a filter with a suitable frequency response is chosen. This is converted into an array of frequency versus amplitude. This example uses a raised-cosine filter with an alpha factor of 0.4. Then the Inverse FFT is computed on the response and the resultant output circularly-shifted to produce the real impulse response. Next, a pseudo-random bit sequence is generated by writing another VBA function. This bit stream is stored in an array of cells. Then the channel impulse response is convolved with the pseudo-random bit stream, and the resultant time-domain signature of the ringing filter is produced as an array. Finally, many pieces of this time signal, each 3-bit times long, and each offset by one bit time are generated as a 2-dimensional matrix. All of the signals in the matrix are then plotted on top of one another, resulting in an eye diagram.

The diagram below show the frequency response of the channel for an alpha=0.4 sinc-compensated channel filter.



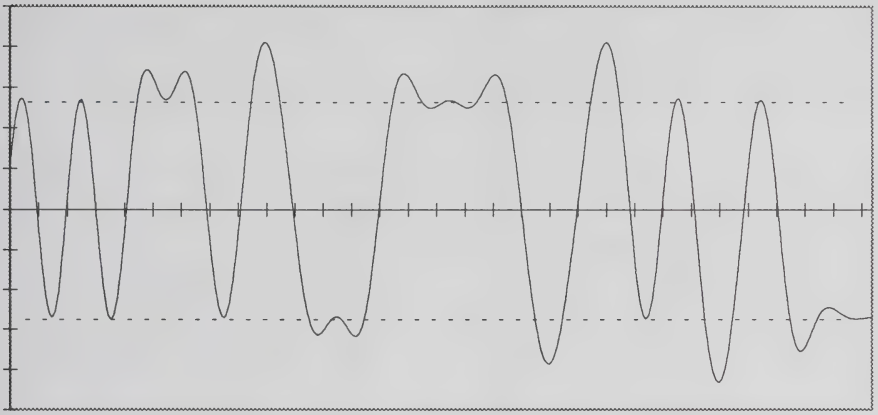
The impulse response of this channel is the computed using the IFFT, and the impulse response is shown below.



Note that the zero-crossings of the sinc-compensated impulse response do not cross through zero at the bit time (an uncompensated raised-cosine impulse response does cross through exactly zero at the bit time) Next, the convolution of the pseudo-random bit stream with the impulse response results in the time-domain waveform from the filter, shown below.

Time-domain response of x/xin(x) compensated raised-cosine filter

alpha = 0.4, sequence = 2^5 - 1 PRBS

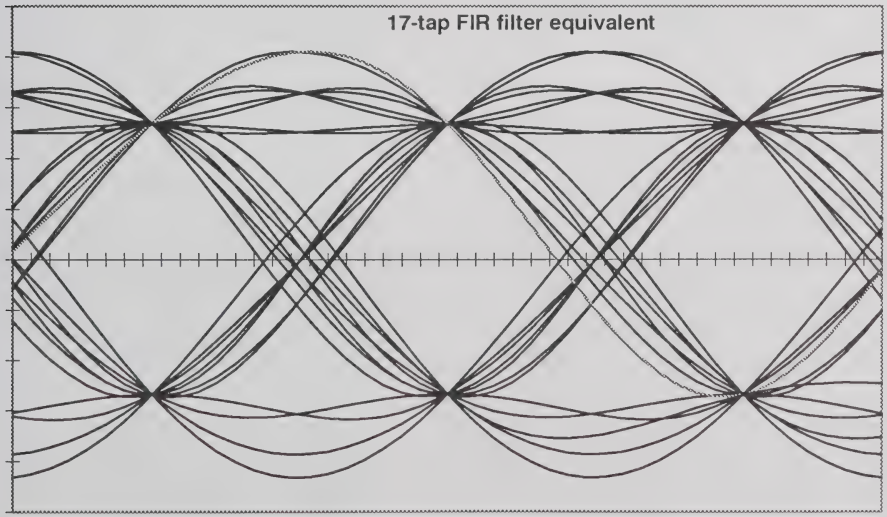


One division equals one bit time

And finally, the resultant eye-pattern from the above time-domain waveform is shown.

Eye Pattern - Raised cosine + x/sin(x) compensation

alpha = 0.4, sequence = 2^5 - 1 PRBS



Summary

It is possible to graphically solve many interesting problems in the design and analysis of data communications systems using spreadsheets. Three examples have been presented which illustrate the utility of this method.

Building a Packet Network

by Karl Medcalf WK5M

Background

Since the beginning of amateur packet radio, users have tried to push the limits. This has taken many forms: how far can I get, how much data can I pass, how fast can I go.

In 1987, Software 2000, Inc. developed the NET/ROM™ code, which replaced the EPROM in TAPR clone TNCs, in an attempt to improve the packet situation. This code provided the first attempt to build a network using amateur packet radio. Much of the current network system throughout the world is based on this code, and new implementations continue to arrive on the scene.

Purpose

This paper will present various viewpoints on network construction, and does not intend to imply that any one concept is superior to any other. It is intended to provide node operators (current and future) with ideas for consideration to help improve the existing system.

We'll first present the network as it exists in many areas, mostly at 1200 baud on both user ports and backbone ports, and then present ideas for expanding and modifying the existing systems to provide higher throughput and reliability.

Network Concepts

Packet radio networks are intended to provide users with the necessary means to pass data from point A to point B with little regard for the details involved. As such the network node operators must be aware of the impact that any changes and additions to the network may cause.

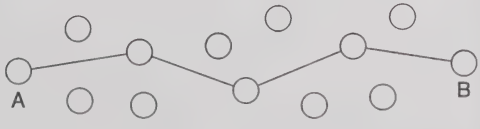
When amateur packet radio began, each packet station was virtually an independent entity. Users could connect to nearby stations and pass data unencumbered, but long distances could not be spanned easily. Every packet station had the ability to be a digipeater, a digital relay station that could retransmit whatever packets it heard, thus extending the range of communication. In order to use these digipeaters, however, the user had to know the callsign and/or alias of each digipeater from the beginning to the end of the route, and also had to know what order the digipeaters had to be used.

As we look at figure 1, we see how a user might connect to a distant station using these digipeaters. This system, although it was used successfully, had many limitations. In order for this to work, each station in the link had to be able to communicate directly with its neighboring station, and since the AX.25 protocol places a limit of 8 relay stations, this limits the potential range. In addition, each station in the link must be turned on and operating – this could be hit and miss since some people tend to turn their systems off when they aren't operating.

Another defect with this system is that once the link is established through these relay stations, failure of any one station would effectively cause the complete failure of the link. This failure could be due to natural causes (lightning strike, power failure) or through human causes (the user turned off his TNC or changed frequencies on his radio).

Although figure 1 shows that there may be more than one path from point A to point B, the packet system has no means to select the "best" or most reliable path, nor is it capable of detecting failures and routing the data around the failure. With the release by Software 2000 of the NET/ROM firmware, a new era of packet radio began.

FIGURE 1



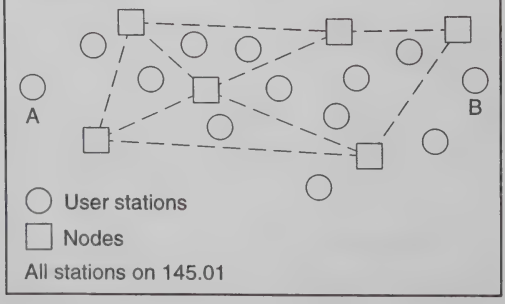
The Beginning of Packet Networks

By replacing the EPROM firmware in the relay stations, these stations became intelligent for routing data from point A to point B. The new firmware converts these TNCs from simple digipeaters into Network "nodes". The nodes broadcast a simple packet on a periodic basis, typically once each hour, and thus other nodes recognize the presence of neighboring nodes automatically. Each node maintains a table of all of the other nodes it can hear (its neighbor nodes), and broadcasts a list of these nodes. Each node would listen to the neighboring node broadcasts, and thus learn of distant nodes – nodes which cannot be heard directly, but which the neighbor can hear. For instance, if Node A can talk to Node B, and Node B can talk to Node C, then Node A can talk to Node C (by relaying through Node B).

The initial network was built using existing 1200 baud TNCs connected to 2-meter radios, and operating on the same frequency that packet operators had been using for some 5 years – 145.01 MHz. This provided a relatively inexpensive means to improve the packet system – no new investment in TNCs, radios, or other hardware was required – only a new EPROM. As we look at figure 2, we see how the network may have looked in the beginning. All stations were operating on a single frequency, and all were operating at 1200 baud.

Problems quickly arose with this system. Since all nodes and all users were on the same frequency, the frequency quickly became overloaded with data. Remember, each node broadcasts its list of other known nodes periodically and the users were still trying to use the same frequency. Additionally, many nodes were installed at relatively high altitudes enabling them to talk greater distances, but the additional height added to the hidden station problem. A user attempting to connect to a node may suffer collisions from a neighboring node many miles away.

FIGURE 2



A Better Way

To overcome some of these problems, node operators began building node "stacks" – two or more nodes at the same physical location connected to each other through the serial port of the TNCs. These node stacks would have one node dedicated to "user" access, and the second node, on a different frequency, for node-to-node or "backbone" communication. This scheme allowed the data to pass with little contention over the backbone, and also reduced the hidden station problem since local users were no longer sharing the frequency with distant nodes. As more users and more nodes began to appear, node operators extended the node stacks with additional TNCs. The additional nodes were on different frequencies or bands, and in some cases operating at higher speeds.

High speed backbone systems are seen as one of the keys to success of a network system. Unfortunately, at that time, 9600 baud packet radio was in its infancy, and there were no commercially available radios capable of operating these speeds. Node operators, in their quest for higher speeds, modified existing radios to permit the higher speeds. Today, 9600 baud is rapidly being introduced into networks – partly because the current system is severely overloaded, and partly because the major radio manufacturers now produce radios capable of operating 9600 baud without modification.

One Network Plan

Figure 3 shows one possible plan for developing a network. Each node in the system consists of at least 2 ports – one for the Local Area Network (LAN) to provide user access, and a second for the backbone. The user access port is a 1200 baud port, while the backbone port typically runs at higher speeds –

FIGURE 3

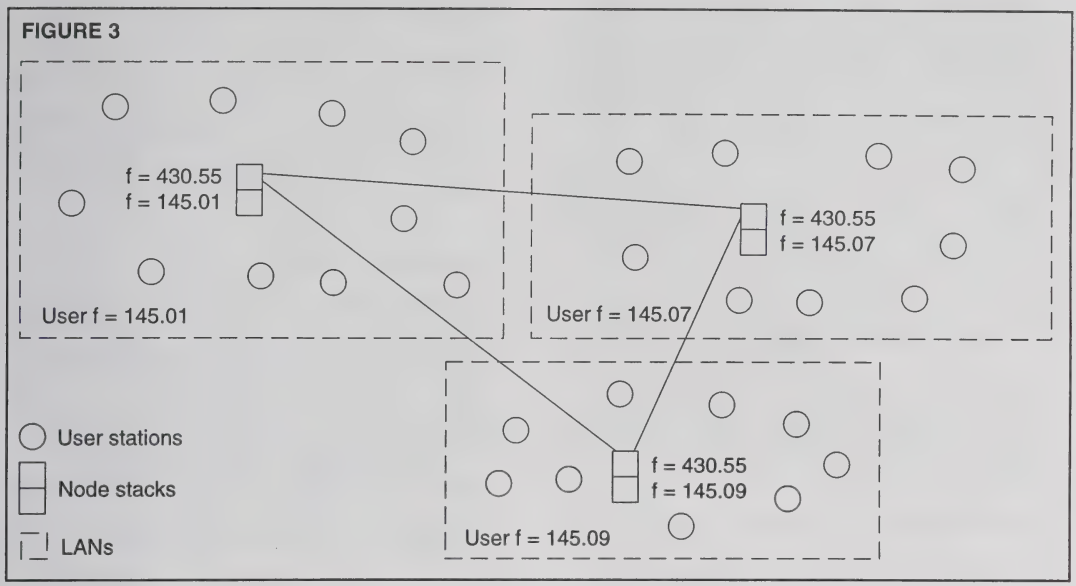
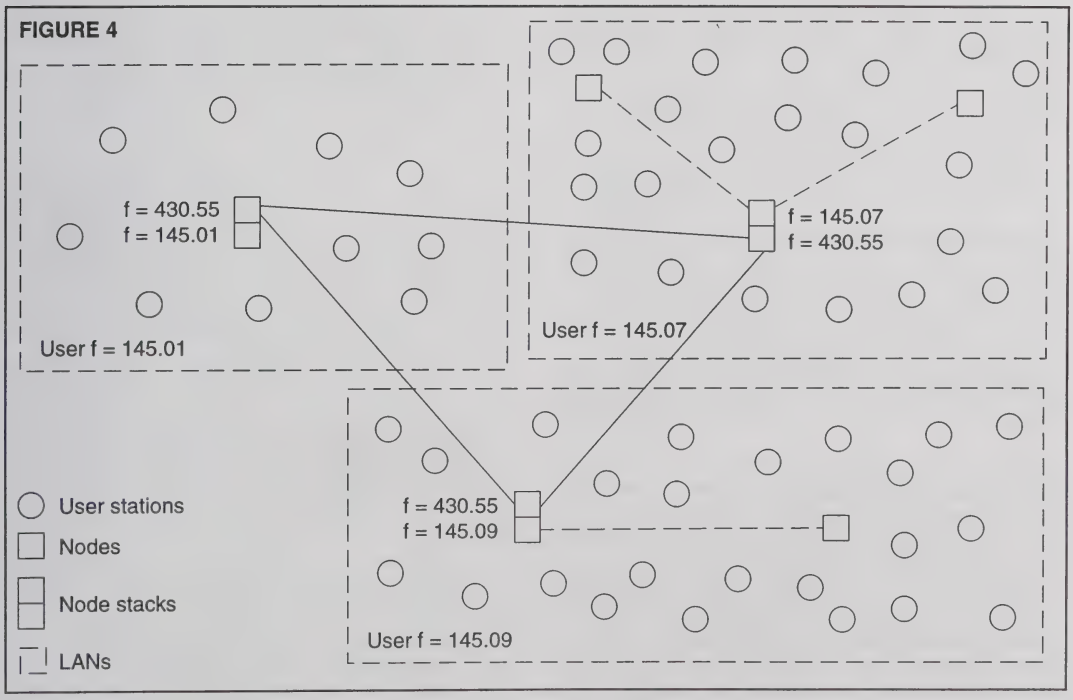


FIGURE 4



9600 baud or higher. Adjacent area LANs operate on different frequencies so they do not interfere with each other. The two ports are connected together at the node to allow users from one LAN to connect to users in another LAN through the backbone.

In this example, the user access port should have its antenna at a relatively low height, perhaps 20 or 30 feet, so that all users in the LAN can reliably connect to the node, but also so that all users in the LAN are close enough that they can hear each other. This reduces the possibility of collisions between users accessing the system. The backbone port antenna is generally located at a fairly high spot, providing the ability to connect to very distant nodes with few intermediate nodes required.

Frequently, users in one LAN may wish to talk to users within the same LAN, and therefore do not require the services of the network. It may well be advisable, therefore, to provide a "direct connect" frequency within the LAN for these purposes. Typically this frequency would not have nodes linked to the backbone system.

In some areas, due to available resources, it may not be feasible to have the LAN cover a small enough area that all users can hear each other. This would normally occur in a lightly populated area where the user base cannot afford to install many dual-port nodes. In these cases, the backbone link may be provided by a single node with two ports, and additional low-speed nodes installed on the LAN frequency to increase the LAN coverage area. Figure 4 shows such a system, expanding figure 3 with additional low-speed nodes.

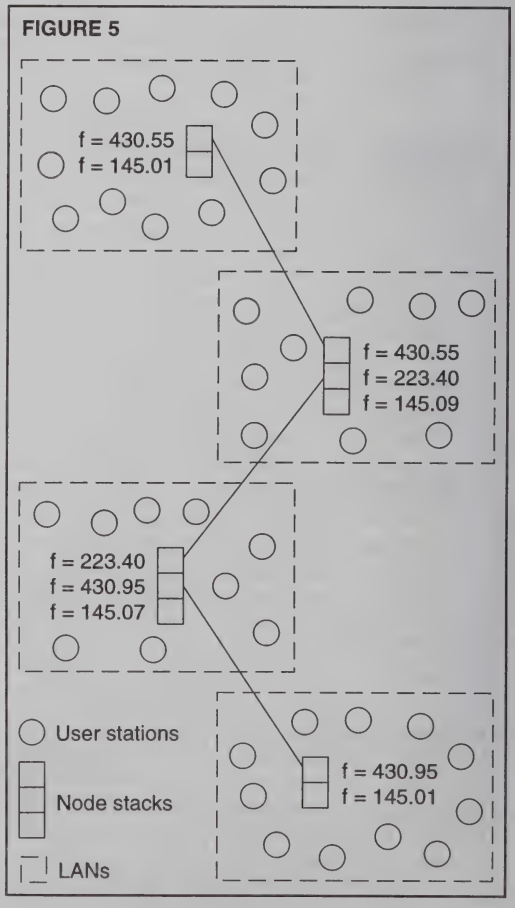
Another possibility

The previous plan can have some drawbacks. With the backbone nodes being capable of hearing very distant nodes, there is the possibility of collisions between the nodes which could severely impact throughput. To overcome this problem, some areas are using staggered frequencies and/or full-duplex between the nodes. Figure 5 shows a possible configuration using this scheme. This scheme requires that each node be equipped with directional antennas, and must be capable of transmitting and receiving on at least two frequencies for the backbone, plus whatever user frequencies are required. Obviously this can quickly become a cost-limiting factor, but in heavily populated areas the cost may be shared by many users.

Network Loading

Users generally do not present a heavy load to the network system. Since users typically type slowly, and send short messages to the station they are communicating with, the high-speed backbone is very capable of handling several users without problems. However, Bulletin Board Systems (BBSs), DX Packet Cluster Systems, Conference Bridges

FIGURE 5



and the like present a very different situation to the network.

BBSs provide users with a tremendous amount of information – regardless of what you may think of the value of that information. This data arrives at the BBS from another BBS and so on, which means that the data has to travel through the network at some point. Within a LAN, users access the BBS to retrieve information, but there may be only one BBS in that LAN. Therefore the BBS must use the backbone to send data to other BBSs. This presents a couple of questions to consider: 1) should users access the BBS on the LAN frequency, or should a separate frequency be assigned for user access to the BBS services, and 2) should the BBS forward traffic through the network via the low speed (user) access port of the node, or should the BBS have direct access to the backbone frequency.

There are no concrete answers to these questions, they must be decided on a local basis. Things to consider include the channel loading caused by users accessing the BBS, the cost of the equipment required, available frequencies in the area, and perhaps other "political" factors.

Node Ownership

When networks first started, all that was required was a simple TNC, some new firmware, one radio, and a two-meter antenna. The overall investment to set up a node was relatively small. Those who built the network already had a TNC, and the additional investment for a used radio, small antenna, and cheap power supply was no problem. However, as the network grew, node stacks started appearing, and the cost of multi-port nodes grows quickly. For example, a single two-port node might consist of

1 – 1200 baud packet TNC	$130
1 – 9600 baud packet TNC	$200
1 – 2-meter radio	$300
1 – 2-meter antenna	$50
1 – 70-cm radio	$350
1 – 70-cm antenna	$75
2 – power supplies	$150
Feed line	$250

This brings a simple two-port node to a minimum cost around $1500. Not very many individuals will spend this amount to provide a hilltop node, and if there is a need for a 3- or 4-port node, the cost becomes prohibitive. If you are lucky enough to find a person willing to provide such a node for your system, you're very fortunate, indeed. Consider, however, that private ownership of the nodes comes with potential problems: what if the owner dies, or gets tired of packet, or needs the cash from selling the gear, or moves? You may suddenly find a hole in your network that you can't easily replace.

Perhaps a better solution is for packet groups to sponsor nodes within the area. In such cases, the packet group owns and operates the nodes, so one member of the group moving or tiring of the mode doesn't disrupt the network.

Network Parameters

So now that you've built a network, how does it automatically select the best path from point A to point B? The answer lies in the parameter settings within each node. Network nodes listen for the node broadcast from all neighboring nodes, assign a "quality" to each neighbor, and calculate a quality to each destination the neighbor has reported. These quality figures are then used in the routing of packet data to its final destination.

There are probably as many theories about setting network parameters as there are network node operators. This paper describes one such idea that is being used in the Kansas network and seems to provide reliable node operation.

Quality figures can range from 0 to 255, with 255 being "perfect" and 0 being totally unusable. As we set the quality to a neighbor, there are several items to consider. First, what speed is the link to this neighbor. A 9600 baud link is better than 1200 baud, but 56 KB is even better. If the backbone (link) frequency is on 2-meters and users are also allowed on this frequency, this is not as good as a 2-meter link that is closed to users. Perhaps the link is on 70-cm, which typically has fewer users and therefore should be a higher

quality. What is the distance between nodes, and how reliable are the connections between these stations (don't forget to account for varying weather conditions here). A link may be 100% reliable during the winter months when there are no leaves on the trees, and fail miserably when the leaves come out. Rain and high humidity can also affect the reliability of a link.

Given the above, we developed the following guidelines (this is only a portion of the guidelines):

1. A 70-cm link with high-reliability, no users, and operating at 19,200 baud will be assigned a quality of 220.
2. A 70-cm link with high-reliability, no users and operating at 9600 baud is assigned a 200 quality.
3. A 2-meter link with high-reliability, users allowed and operating at 1200 baud is assigned a 140 quality.

These high-reliability neighbors are "locked in" at the quality shown, and the node is configured so that other nodes heard on these frequencies are assigned 1/2 the quality of the locked in nodes. Figure 6 shows this scheme and the resulting quality for each node.

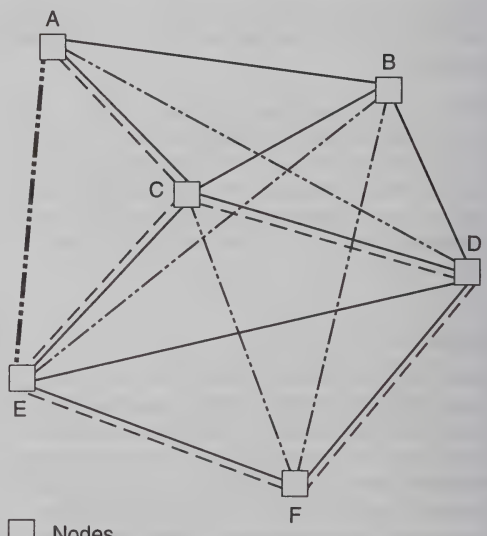
Using this scheme, during band openings, the distant nodes that are heard are assigned a relatively low quality, causing the node to continue to use a more reliable path whenever possible. Although this will tend to limit the number of nodes in the nodes list, a user attempting to connect to one of the distant nodes will probably succeed.

Building A Network

Since the release of NET/ROM in the late 1980's, there have been many derivatives of the networking protocol. The first of these was TheNet, introduced by NORD><LINK. Other derivatives that followed were the G8BPQ code, developed by John Wiseman, TheNet Plus, various flavors of TheNet X1-J, most TCP/IP NOS programs, and now Kantronics has introduced K-Net™. All of these systems use the same networking protocol, so your network may consist of a mixture of the various types of nodes. Some of these may offer features not found in others, but the basic networking protocol remains constant.

In the past, installing a node meant that you had to remove the standard user firmware (EPROM) in your TNC and replace it with one that was specifically written for networking. This meant that you gave up the "private" use of your TNC and dedicated it to being a node. In addition, this required you to burn your own EPROM after modifying certain bytes in the image to contain your call-sign, node alias, and your chosen parameters.

FIGURE 6



Quality from Node A to Node:	Normal Conditions	Band Opening
B	220/189	
C	220/189/140	
D	189/120	70
E	189/120	140
F	162/104	95

To create a two-port node, you needed two TNCs, each equipped with the special EPROM (but each EPROM needed its own callsign and alias). These two TNCs were connected together through their serial ports. As you expanded to a three- or four-port node, more callsigns/aliases were required, and connecting the serial ports together required a somewhat complex diode matrix to avoid data collisions on the serial ports. The number of diodes required is determined by the formula 2*N*(N-1) where N is the number of TNCs being connected together. For a three-port node, that's 12 diodes and for a four-port node you need 24 diodes.

With the release of Kantronics K-Net for the KPC-9612, users now have the ability to build a two-port node with only one TNC, providing both a 1200 baud user-access port and a 9600 (or even 19,200) baud backbone port. Since both ports reside in the KPC-9612, the node requires only one callsign and one alias. The unique features of the K-Net in the KPC-9612 don't stop here though – the TNC can still function as an end-user TNC. The PBBS in the KPC-9612 is still usable (there's a BBS command on the K-Net node) and a user can still run multiple connects, a KA-Node™, and even operate Host mode with specialized software.

Configuring a K-Net node is unlike other nodes – all the necessary commands are available at the command prompt. There is no need to patch the EPROM with the callsign and alias as other nodes require. Couple this with the remote access capability and you can place the K-Net/KPC-9612 on a remote hilltop. When the current trustee moves, an authorized user can connect remotely and change the callsign and alias of the node without making a trip to the site. In addition, the K-Net supports the NET/ROM interface over the RS-232 serial port, so it can be easily connected to an existing node stack using other brands of TNCs.

Building a four-port node with two 1200 baud ports and two 9600 baud ports would simply require two KPC-9612s with the K-Net firmware, and an interconnecting RS-232 cable with only three wires – ground, TXD, and RXD (TXD and RXD must be crossed). If you only require one high-speed port but need two 1200 baud user ports, that can be accomplished using a KPC-9612 and a KPC-3 with the optional K-Net firmware EPROMS.

Building or expanding network systems is not extremely difficult, but for the network to operate smoothly and provide maximum benefit, it requires cooperation and coordination between the node operators, BBS system operators, and other network service providers.

DAMA – Another approach

There has been some discussion and previous papers presented on DAMA (Demand Assigned Multiple Access). While this sounds like a completely different networking scheme, it really isn't. The basic premise of DAMA is twofold: 1) Prevent collisions between stations using the node, and 2) Allow additional time for stations sending more data than for those sending little data.

A DAMA node (called a DAMA MASTER) is actually a LAN node with at least one link to a backbone network. A single DAMA master can support up to 16 TNCs (thus 16 radios), perhaps providing two or three backbone link frequencies, a couple of user frequencies, some frequencies for BBS systems, and so on. Each LAN has a DAMA master node, and the adjacent DAMA masters are linked together. This inter-node link uses NET/ROM protocol, exactly like the networks just discussed.

Within a LAN, users connect to the DAMA master. Through special coding in the AX.25 frame, the user's TNC is placed into a "DAMA SLAVE" mode. From this point on, the user's TNC will not transmit any packets until the master has instructed it to do so by sending a poll frame to the slave. In this manner, no two slave TNCs will transmit at the same time, eliminating collisions.

The demand access feature of DAMA comes through intelligence built in to the master node. When the master polls a user, the user must respond – even if it's only an ack saying "I have no data to send". After polling station A, the master polls station B, then C, and so on. Any station that does not have data to

send has its priority reduced. The master then polls those stations with higher priority more frequently than those with low priority. As soon as a low priority station responds to the poll by sending data, the master bumps that user's priority back up to the maximum.

The DAMA system may provide an excellent alternative in areas where LANs are very large, resulting in many LAN users being unable to hear all other users. The major drawback to the DAMA system, at this point, is that the DAMA master station requires a computer at the node site to run the special software. In addition, multiple frequency operation requires one TNC for each frequency with special DAMA firmware installed.

Users accessing DAMA systems should be using special DAMA slave TNCs. The DAMA software as distributed contains EPROM images for TNC-2 clones, enabling them to be DAMA slaves. These EPROMS use a slightly modified version of the WA8DED Host mode to communicate to the computer, and some software packages are readily available for use with these. Among the more popular programs in this category are Grafik Packet and ESKAY.

Kantronics will soon be releasing a new DAMA capable EPROM for some models of their TNCs. The new DAMA EPROM will allow users access to DAMA networks as a slave using any terminal program, dumb terminal, or even Kantronics Host mode.

Conclusion

Networking in the amateur packet world can be considered to be still in its infancy. The first NET/ROM firmware was released around 1987, and many improvements have followed. Other networking systems have been developed (ROSE and TCP/IP) for amateur use, but to-date, none has been declared the clear "winner" for packet networking.

Debates rage about the "best" or "perfect" network, but if such a system already existed, world-wide VHF packet would be a reality instead of a dream.

DAMA – Another Network Solution

by Karl Medcalf WK5M

Background

Amateur packet radio as we know it started in the early 1980's, and since that time, packeteers have been searching for better ways to expand the range and usefulness of packet. Among the developments that have assisted in this are the many bulletin board systems (BBSs), the DX Packet Cluster system, Chat nodes, NET/ROM™ and compatible networking code, ROSE packet switches, and others.

Purpose

This paper will discuss some of the drawbacks to the most widely used networking system (NET/ROM and its derivatives), and one idea to help alleviate some of the problems. It will explain, in some detail, the DAMA (Demand Assigned Multiple Access) protocol as proposed and implemented in some areas of the world.

Network Problems

Anyone who has used the packet radio network in place today has noticed problems, and the range of problems varies with the area of the country, the type of nodes being used, channel loading, and other factors. Types of complaints commonly heard cover the spectrum from simply "I can't get anywhere through the network" to "it's just too slow" to "I need 200 watts just to connect to the node".

These problems are currently being addressed by network node operators and interested packet organizations around the world. Many areas have regular node sysop meetings to discuss the connectivity of the network and means to improve it. These discussions usually center on proper set-up of the node parameters, providing users with a network that actually lets them connect across the network. To this end, the Northeast Digital Association has implemented a World Wide Web page on the Internet with information about their recommendations on node parameters (http://www.cam.org/~dino/neda/neda.html).

It is quite possible that newsgroups may appear for this subject, as well as home pages from other organizations interested in the growth of packet.

The speed problem is also being addressed. Most TNC manufacturers have been producing units capable of higher speeds (9600 and 19,200 baud in particular) for quite some time, but the readily available commercial radios were not capable of operation at these speeds. Within the past year or so, however, major radio manufacturers have started producing equipment that is ready for operation at 9600 baud without modification. Once implemented in the network, these higher speeds will become apparent to the users, and the data throughput figures will increase.

The third problem is generally caused by collisions of packets transmitted by two or more users. Most often this is caused by the node being located at such an altitude that it can hear users over a wide area. Unfortunately, all the users cannot hear each other, and thus collisions are a frequent problem. Since FM is used as the modulation scheme, the FM capture effect simply says "the strongest station wins!" Thus some users are increasing their own power in order to use the node, but this causes more collisions, causing other stations to increase their power. This creates a never ending circle of guaranteed network problems.

Over the years, many suggestions and papers have been presented with ideas for solving the network problems. Some suggested new network protocols, others have been attempts to change user attitudes and operating practices.

The DAMA solution

In recent years, a group of Germans (NORD><LINK) has developed a system called DAMA – Demand Assigned Multiple Access – in an attempt to address the collision problem within a network. The DAMA system consists of one node (called a DAMA master) in each LAN (Local Area Network). The DAMA master nodes are then connected to

each other using the standard NET/ROM type protocol.

Within the LAN, users operate specially equipped TNCs which contain firmware (EPROM) that can operate in a DAMA slave mode. These TNCs will normally operate CSMA (Carrier Sense Multiple Access), but are switched into the DAMA slave mode whenever they have an active connection to a DAMA master. In the slave mode, the user's TNC will not transmit any packets until the master has granted permission through a polling system.

The unique ability to reduce and/or eliminate collisions in a DAMA system is provided by the polling concept used in DAMA. A user connects to a DAMA node with a normal AX.25 connect frame. When the DAMA master responds with the UA frame, however, the SSID field of the DAMA master's callsign has one of the reserved bits set to 0, indicating DAMA mode is being used. The special user TNC recognizes this and the user is placed, automatically, into a DAMA slave mode.

Once placed in the slave mode, a user's TNC will not transmit any packets until it has received this poll from the DAMA master. The poll to a user consists of a frame from the master station to the slave. This frame may be a simple poll frame, or even an information frame from another user to the slave. (All frames transmitted by the master have the special DAMA bit set to 0, indicating DAMA.) Anytime the slave receives a frame from the master, it may then transmit all frames it has prepared. Since the master has only polled one station, no other stations in the LAN will transmit, eliminating collisions.

The polling from the master to the slaves occurs in a round-robin fashion. The master first polls station A. If station A has nothing to send, it responds with an RR frame as a response, otherwise it sends all pending frames. The master then polls the next station (station B). The master does not acknowledge the frames that have been sent by station A until it has polled all other stations in the LAN. Initially, this allows all users equal access to the node.

Equal access may not, however, provide the most effective channel usage. If one station is sending lots of information (perhaps a BBS) and another station is only sending an occasional data frame, then much of the time spent polling the occasional user results in wasted spectrum. To avoid this, DAMA assigns a priority to each user. When a user first connects to the master, it is assigned the highest priority, and is polled each time through the list of connected stations.

When a polled station responds with only an RR frame (no data to be sent), the master reduces that station's priority, and skips that station on the next polling sequence. As continued empty responses are received from that station, the priority continues to reduce to a minimum value. In this manner, those stations that are sending information frames are polled more frequently than idle stations.

When a station with reduced priority responds to a poll with information frame(s), the node bumps the priority of that station back to the maximum value. Thus users who sit idle for long periods and then start sending large amounts of data are not penalized for their inactivity.

DAMA slave stations retain the ability to connect to other DAMA slave stations, and in many cases this may be advisable. If you're located very close to a neighbor and want to connect to him, your TNC can connect to his in the normal CSMA method we use today. This allows for better throughput between the connected stations, but can present collisions to users requiring support of the DAMA master.

One side effect of this direct connect is that if either DAMA-equipped TNC connects to the DAMA master (or if the master connects to either TNC), that TNC will suddenly enter the DAMA slave mode. The slave mode station will not respond to any frames (even those from the neighbor) until it has received permission from the master, but the neighbor station will continue to use CSMA. This can result in a time-out from retries since one station is DAMA slave and the other is CSMA.

Why Use DAMA

The DAMA system can drastically reduce collisions between users in a LAN. This occurs most frequently when a node is installed at a relatively high location to provide coverage to a wide area. In these cases, installation of a DAMA master will improve user access while maintaining connectivity with NET/ROM or equivalent nodes.

There are several factors that can influence your decision to install a DAMA node. Among these are the number of users trying to access the node, the amount of area covered by the node, and the cost of installing a DAMA master.

With one or two users in a given LAN, a DAMA system may not be required since the probability of collisions would be rather small. However if the LAN has 6 or 7 users all operating at the same time, perhaps the DAMA node becomes cost effective.

The amount of area covered by a node can be controlled by several factors. First, it's possible that regulatory requirements limit the ability to install nodes. Perhaps you must have a special (maybe very costly) license to operate a node. In this case, very few nodes may be installed, and they must be located as high as possible to cover the area. Another possibility is that within a large area (say western Kansas) you only have a few users, and therefore one node can easily handle all of the traffic.

The third consideration is cost. Installing a DAMA master requires a computer running specialized DAMA software called TheNet NODE (TNN) and one TNC with special EPROM for each radio frequency on the system. Although the price of used PC computers is coming down, a DAMA master should be operated on a relatively fast computer to service all of the TNCs and users in a timely manner.

Building a DAMA System

The DAMA system actually consists of two major parts: the DAMA master (node) and DAMA slaves (users). The DAMA master consists of a PC compatible computer (286 or higher is recommended) connected to one or more TNCs. The PC is the master control of the entire DAMA system and runs the TNN software. Initial setup of the TNN software can be difficult, as the documentation is written in German. The TNN software (source code available) is compiled to support the configuration you need – the version we have was compiled for one COM port supporting 16 TNCs. The TNCs that are connected to the PC running TNN software must be running special DAMA EPROMS. These EPROMS basically contain the WA8DED firmware which has been modified to support the DAMA protocol.

Cabling the TNCs to the computer is slightly different than normal, as the TNN software uses a token ring arrangement to poll each of the TNCs for data. The data from the computer (TXD) is connected to the first TNC's TXD pin. From this TNC, the RXD pin is connected to the TXD pin of the second TNC, the RXD from the second TNC is connected to TXD of the third TNC, and so on. The last TNC in the ring has its RXD pin connected back to the RXD pin on the PC running the TNN software.

Because of this ring arrangement, any data received by the first TNC must be sent completely around the ring through all other TNCs before reaching the PC. Likewise, any data to be transmitted by the last TNC must first be relayed through all preceding TNCs. For this reason, NORD><LINK recommends the serial port be operated as fast as possible, and the TNN software supports token-ring speeds from 9600 baud through 115,200 baud.

The EPROM installed in the token-ring TNCs must be configured separately for each TNC. Since they are installed in a ring, each TNC needs its own address – a single byte in the EPROM defines this. In addition, you may have a DAMA node in which some TNCs are supporting DAMA users, and others may be required for links to other nodes and/or users who do not have DAMA capability. Therefore one byte in the EPROM image enables DAMA mode on the TNC.

Users also have special requirements for their systems. Although it is possible to use any TNC to communicate with a DAMA system, using the standard CSMA system can severely impact the throughput of a DAMA configuration. Ideally, all users within the DAMA network should be configured as DAMA slaves. To accomplish this requires one of three approaches. A special EPROM image has been made available to replace the existing EPROM in TNC-2 clone TNCs, making them capable of DAMA slave operation. These EPROMS can use "terminal mode" from the TNC to the computer, or they can be switched into "host mode" which enables the WA8DED host firmware between the TNC and computer. Using this host mode it is possible to run specialized programs such as Grafik Packet, SP, or ESKAY.

The second possibility is to use any KISS mode TNC and a TSR (terminate and stay resident) program in your PC computer. The TSR (called TFKISS) communicates to the TNC using the standard KISS protocol, and provides a WA8DED interface to a terminal program in your PC. Again this allows you to use the Grafik Packet and other similar software.

The third option requires that your TNC manufacturer provide special EPROMS which have support for the DAMA slave mode included. Kantronics has recently announced such EPROMS for the KPC-9612, KPC-3, and KAM Plus TNCs. With this system, you can use a generic terminal program (e.g. Procomm Plus) or Host mode programs such as Host Master II+, KA-Gold, KA-Win, or XP-KAM. The main advantage is the ability to use familiar software, and in the case of the KAM Plus, you can even operate HF modes (i.e. G-TOR, Pactor, RTTY) while your VHF port operates DAMA slave.

Conclusion

Network construction is the future of packet radio. Unless the network can be developed to the point that users can easily accomplish their needs, the network will never be seen as performing properly. The only way such a network can be implemented is for the node operators to be open to new ideas and concepts, and to effectively communicate and coordinate with other node operators.

Unfortunately, there is no national organization that oversees network implementation, and many argue that there shouldn't be. However, without adequate coordination between areas, the network may well be destined to its current state – some places it works, and others it doesn't.

DAMA may well contain some solutions to problems experienced in parts of the network, and may be detrimental in other areas. The perfect network doesn't exist (yet), and perhaps DAMA will prove to be beneficial in your area.

The Tulsa National Weather Service TexNet Interface Project

Bob Morgan, WB5AOH (morganb@tenet.edu)
Greg Jones, WD5IVD (wd5ivd@tapr.org)

Texas Packet Radio Society • P. O. Box 50238 • Denton, Texas • 76206-0238

Abstract

This paper details information concerning the interface to the NWS system in Tulsa by the TexNet network and how it can be replicated by other networking systems at other NWS sites. Several functions are provided from the TULSWX node including: automated severe weather alert broadcasts, emergency service broadcast, color weather radar images using the NexRad Doppler (the WSR-88D), and the dissemination of messages directly to various EOCs and spotting groups.

Introduction

The NWS TexNet Interface project began in 1992 and went on-line in 1993. It began with amateur radio Skywarn spotters in Tulsa working in conjunction with NWS forecasters to examine ways to add packet radio operations to the NWS location in a meaningful manner. The Tulsa Skywarn organization has for many years been very close to the national weather service personnel.

For the last several years, the NWS has undergone a program of consolidation where many of their forecast offices have been moved and closed. Some of the burden of communicating weather observations to and from the public is falling to amateur volunteers. Packet is providing an important link between the NWS and the amateur volunteers. Voice reports from spotters are still as necessary as they ever were, even with the newer radars, because neither is capable of observing the same information set. Packet finds its place in communicating directly to various emergency nets and local EOCs that are spread out over ever increasing distances, as NWS sites are consolidated. These new distances make voice communications, once the mainstay of communications, nearly impossible for the NWS to use to communicate directly over these larger geographic areas.

TexNet, since its inception, was designed with emergency weather net operation in mind, partly because of its geographical location within Tornado Alley. When the Tulsa Skywarn organization made their wishes known for a packet system, TPRS was asked to provide information on its capabilities. A few of the Tulsa amateurs were familiar with the DFW area weather PMS (Packet Message System), and even at a distance of some 300 miles had used it a few times and knew what it was capable of providing.

This led to the creation of the TULWX node. A standard TexNet node is equipped with harddrive and a weather feed provided by the NWS. The NWS allowed the

equipment to be located on site, which provides secure facilities and has AC power backup. The only site drawback is there is no 2m user port; thus, users have to connect to the Tulsa weather node using nodes elsewhere on the network. The NWS personnel, including their manager, have told us they are almost as proud of the capabilities of the PMS to distribute vital weather information to EOCs and spotters in a non-time consuming manner as they are of their brand new WSR-88D Doppler radar, and of course, the weather node is a whole lot cheaper.

The PMS, it should be noted, is a 10 user Packet Message Server, which is implemented on a TexNet node which contains a disk controller and drive. It has been included in TexNet almost since 1986, as has some form of weather server on the TexNet network. The original TexNet weather server was interfaced to a 5-bit baudot coded landline teletype circuit, which is now a thing of the past. A weather server is a standard PMS which includes an async serial capture port and a search database that relates product codes to a fixed set of message numbers. It is usable as any PMS to save and read messages and also to read messages stored from the NWS input. Messages are available within seconds from the time the message is released by the forecaster.

Several functions are provided from the TULSWX node. These include: automated severe weather alert broadcasts, emergency service broadcasts, color weather radar images using the NexRad Doppler (the WSR-88D), and the dissemination of messages directly to various EOCs and spotting groups.

Automated Severe Weather Alert Broadcasts

This function is the newest showpiece of the weather server application of the TexNet 1.72 revision, according to the beta-testers in Eastern Oklahoma, where it is running on TULWX and two Muskogee nodes. When any type "S" (Severe) weather message is stored to the disk of the PMS, the message heading and the first 148 characters (following the NWS Product ID Code) are broadcast all over the 1.72 portion of the TexNet network -- to all ports of these nodes. The broadcast to the intended users is from an alias of "ARES" to another alias of "WX" as a UI frame. ARES stations, EOCs and the like set their TNCs to include "ARES" in their budlist and they enable the terminal bell. The message rings the terminal bell, sends the whole heading of the message, and enough of the first part of the message to display the NWS county warning data and expiration data. This can enable users, possibly by automated means, to log onto the PMS and read the message of interest, and know beforehand whether or not they are interested in it, thereby reducing the PMS user loading while speeding up delivery. This theory is of course contingent on proper administration to discriminate against reading unwanted messages so they don't overload the PMS all at once. Some of the users around Tulsa have written applications which scan monitored UI frames, and when they see one of these, connect to the PMS and automatically retrieve the whole message, and are then capable of taking further action on it. This opens up some interesting possibilities for alerting individuals.

As with all UI transmissions on packet, RECEPTION IS NOT GUARANTEED. Use of this capability is to enhance emergency operating procedures. Users know NOT to solely depend on this feature.

Emergency Service Broadcasts

The Weather Alert broadcast may also be initiated manually by an Emergency Net Control station who has TexNet Sysop privileges. The sysop can then issue an Emergency Service Alert broadcast directly from their keyboard. The same broadcast mechanism is used as found in the Weather Alert broadcast. It is used to alert the ARES packet community of emergency operations or nets being initiated.

A manual command (admin permission required) has been added to the admin command list of the TexNet node to initiate a broadcast. A fixed 228 byte line is retransmitted. Syntax is as follows:

```
A :Enter the text string to be broadcast,
   up to 227 characters and c/r.
```

This is a variation of the admin A -ON - OFF TexNet command. The colon indicates that any text following the colon will be broadcast all over the place. The local outputs are preceded by a bell and a c/r, to ring a terminal bell and turn up a fresh line on the crt of any station monitoring UI frames. The UI frame address is from ARES to WX, the same as with the PMS alerts. The remote output mechanism does not exist on V1.71 and prior nodes. This command does NOT toggle the alert status of the network.

This function was discussed at a spring 94 meeting at Tulsa, needed by severe weather spotters. The anticipated downside of this will be that there is a possibility that many users might connect to the same PMS at once and try to read the same message at once, which has been known to lock it up because of lack of memory.

Color Weather Radar Distribution over Packet

One of the products that the NWS NexRad Doppler (the WSR-88D) system makes available is the Radar Coded Message (RCM). It is an ASCII "text" file, no more than 4K, which is sent by the NWS as a normal weather product that is automatically generated by the WSR-88D (NexRad) systems. Figure 1 shows a black and white print of the color screen image (the image does lose some of its effectiveness in black and white). The text output, shown in Figure 2, can be decoded by software written by Larry Hinson, of the Tulsa NWS office. This program was in fact written to take advantage of the connection that TexNet has with the Tulsa NWS office. The software can be run on any PC and allows the RCM file to be displayed as a lower resolution version of what they see on their own radar user terminal (called a Pup). The RCM output was, up until then, thought of as a low resolution and useless product that happened to exist in the software support of the WSR-88D, until it was discovered it would easily fit on a PMS message track. Since the spring of 1994, these files have been available on the Tulsa weather PMS, and are popular items for storm spotting weather nets and emergency operations centers in cities and counties within range of the radar. This software is described later in the paper.

FIGURE 1 - Weather Display (B&W display of color screen)

Figure 2 - Example NexRad file sent to the PMS which the user could download and display (as seen in Figure 1)

```
ZCZC OKCRCMTUL LOC
TTAA00 KTUL 010307
/NEXRAA05570109940307UNEDITED.
/MDPCPN/SC0906/NI1507:.
HLP110B10110B1,IIM1101E01D001J,IRA8,IIF801C221E221N,IIG12F1D21M21B
,IIL1B221E21221101C2B1F,JHI8001E21D221C2D1210B1,JIN102I33432331101B
,JJG12E3D210101E,JRO1,JJL1B2B11,JND1B2211,JRP1088,KJE1C3221121E221
B,KJF1G221C21B2B1B,KRN108E,KJG1L22112B1B,KRG1C8G,KJP1J22112B1B,KRD
1D78B788,LHA8,LKA1H2C112B1,LQI1F7D88,LHB810C1,LKB1H2C112B1,LQJ1D22
7D8C,LHK10B1D001G2343112B1D0B1D337C8C,LGP8801E01101H24521D4422001F
Z27D8D,MHA81E01D01E23543321B3B1C221B2B7C878B,MJF1C01E234433211210C
1E22337C8D,MHO10C1D01121D32B32B12101E2B337E8C,MHP10C1D01I224454310
1F233227E8C,NHI1B0B1,NKM1H224B32101E2233237F88,NHN1,NJF10B101101C2
34C321001D23C227F88,NHG1,NKG10C101123255ZB001E2B3B7F8B,NMD1B224422
101F2B3B7F8C,OME13215221H2D337F8C,OJB1,OLF10C3C21I2E7I88,OHC7,OMC1
42442B01F2E37H8B008,OLP3443231221F2F7J8D,PLE143321001J2E7I8F,PLF13
2B,POB1E2C1B7G87B8E78808,PLG1,POG1B2F187K808878C08B7,PHP78,POH1B2D
11787787G8G7B8B,QIA8,QLE11,QNI1E2E7788788778G08878778787,QLF121B0C
1G2B788787F88008,QVN8C78,QLK12,QNG1E212B8E7E8,QWC8C,QMH1101D212127
B808007B8778,QWL88,RHA880C8,RMI1I7C80C78E,RXA8,RHB8808B,RMF1D2127E
8080C8C,RIG8,RMC187I8E,RJP8787B80087H8C0B8,SJI87B87788787H88,SIJ8B
08778C778B7D87788,SIK8B0887808G7C8E,SKH8F08C787E,TKI8E008E78788,TL
B8B0C8G08,TNO8808,TJP88,TMD8,TOD8,ULI8,ULF88,UOB8,UJL8C,VJI8D,VKJ8
,VKG88,VKH88,WKA88.
/MT330:NSC.
/NCEN04:C22NNM000000,C05OMD000000,C98ONC000000,C74JLK000000.
/ENDAA.
/NEXRBB05570109940307.
010C010015,020B045034,030B067036,040B097024,050B124020,060A135016,
070A145012,080A146012,090A169011,100A189011,120A226012,140A243012,
160A271013,180A290015,200A265017,350A234044.
/ENDBB.
/NEXRCC05570109940307.
/NCEN04:C22NNMS010HN,C05OMDS010HN,C98ONCS010HN,C74JLKS010HN.
NNNN
```

The NexRad output is updated several times an hour as a weather product saved on the PMS. It has wind and other information embedded in it, in addition to the basic radar image, such as any Mesocyclones (meso) or Tornadic Vortex Signatures (tvs) which the radar system has analyzed. If you live in Texas and Oklahoma, and have a thunderstorm with a TVS in it, you want to know about it, I guarantee you. That's the tornadic vortex inside a thunderstorm which is one of the signatures that the storm is spawning, or may spawn a tornado. Hail can also be shown. The software for the user's pc includes the geographic background or the static part of the radar image, usually the county map lines and city names. The standard distribution is set up for most of the NexRad sites in the nation. It can operate more than one site, and can merge two sites together to subtract ground clutter. It contains wind information which the user can interact with.

The radar site (INX), shown in Figure 1, is located near Inola, Oklahoma, which is east of Tulsa by approximately thirty miles. The Doppler range of NexRad is 124 miles, and that is also the radius of the RCM display, so it reaches into western Arkansas. Its coverage roughly happens to coincide with the TexNet/HogNet network in NE Oklahoma and NW Arkansas and SW Missouri.

To receive the file, the user connects to the network and then makes a connection to the weather server. Then they can initiate a text capture to a disk file, and read the message from the PMS, close the file, exit the terminal comm program, and bring up GDRCM, and display it. The RCM program will remove the overhead of the disk capture and the PMS heading and normally no ASCII file editing is required. Any reliable terminal packet or modem program with a terminal emulator and capture to disk files should do. Procomm and Yapp both work, and I am sure that there are others which will work equally well.

The following are excerpts from the GDRCM software. These are only excerpts, and the program is continually being improved. Please refer to the program for current and complete information. [1]

GDRCM Introduction

The WSR-88D Radar Coded Message Display Software was designed to decode and graphically display the contents of the RCM message for any WSR-88D radar.

Once the RCM software is installed, reflectivity data, storm information, and wind data within the RCM can be extracted and graphically displayed. The reflectivity data, when color contoured and animated by this software, offers an excellent radar display and pinpoints areas of developing storms. Storm movement and maximum tops are also superimposed on the reflectivity data.

In addition, an interactive hodograph is constructed using the wind data in the RCM. Storm relative helicity values are automatically computed for a storm moving 30 degrees to the right and 75 percent of the mean winds from the surface to 25000 feet MSL. Storm relative helicity values can also be computed using individual storm movements as given within the RCM.

This version offers the following additional features:

1) Handles binary and text formatted RCMs. Text formatted RCMs are necessary for transmission through Ham Packet Radio. See Appendix I for further details on the text formatted RCMs.
2) Handles 9 and 6-letter RCM file IDs. (Refer to installation instructions for the handling of 6-letter file IDs.)
3) County and state maps for the 48 contiguous states.
4) Looping capabilities.
5) Ground-clutter suppression capabilities.

Instructions to loop through stored RCMs and draw a ground track.

3.2 Analyze/Loop through Previously Stored RCMs

Press the F2 key from the main menu to bring up a sub-menu of RRR files in the RCM subdirectory. It will resemble the following, with its own set of function key assignments:

<F1>DISPLAY RCM
<F2>LOOP THROUGH LAST 5 RCMS OR FLAGGED RCMS
<F3>ARCHIVE FLAGGED
<F4>DELETE FLAGGED
<F5>RETURN to MENU

The arrow keys move the cursor across the filenames which are ready for display. The display screen will show up to 40 filenames at once, sorted in descending sequence by month, day, and time. If there are more than 40 files, and you want to get to a file not shown, simply continue to move the cursor to the right or down and the display will automatically scroll to the right.

A file can be selected for display by either pressing the <F1> or <enter> key. Files may be looped by flagging the appropriate files using the space bar or <F3>. A diamond will appear to the right of each selected file. Pressing <F2> will process each of the flagged RCMs and loop through the resulting pictures. If no RCMs have been flagged with a diamond, the latest 5 RCMs (or the number of RCMs specified in the RRR.IN1 file) in the data base will automatically be looped. Striking the space bar on a flagged file will unflag that file. <F5> will return the main menu.

While the RCMs are being looped, a number of options exist in what is being displayed. Most of these options are the same as those described under section 3.3. Other options include:

1) The ability to filter out different intensity levels from the screen while the RCM pictures are being animated. This is accomplished by pressing the numeric digits 1 through 6 on the keyboard for the particular intensities you want to filter. Pressing "0" restores the screen back to the original colors.
2) The ability to track a storm's history of locations. This is done by pressing the "T" key on the keyboard. All storm centroids within the RCMs are given a unique 2-digit ID consisting of numbers or letters of the alphabet. These are labeled in blue. Those storm centroids that maintain continuity will consistently be reported within the RCM at 30 minute intervals. The RCM looping program will then find all of these storm centroids that are consistently reported and will draw red line segments representing the trajectory of each storm centroid. The storm centroid locations along the line segments will be denoted with white "+" signs.

Description of the Display

After pressing <F1> from the main menu, or after an RCM message is decoded, a contoured color analysis of the reflectivity is displayed within the left two-thirds of the screen. Superimposed on this data is a county and state map. If a city file has been created for the radar (see section 6, part A, subsection 1.10), these cities will also be displayed.

The vertical wind profile (knots) will then be displayed on the right side of the screen. Winds within the RCM message are coded at intervals of 1,000 feet up to 10,000 feet, 2,000 feet from 10,000 to 20,000 feet, and 5,000 feet above 20,000 feet (Federal Meteorological Handbook No. 11). Because of the data density at the lower levels, winds below 10,000 feet are plotted at 2,000 foot intervals. This results in a more readable vertical profile of the winds.

The hodograph is in the lower right side of the display. All available winds from the surface to 9000 feet are used to create the hodograph. Each wind is represented by a white circle with line segments (shear vectors) connecting each circle. Cell movement and storm movement, as described earlier, are printed at the top of the hodograph and are depicted within the hodograph by blue and yellow circles, respectively. Rings on the hodograph are 5 knot isotachs.

A summary of the options is shown below. These options are displayed at the top of the screen.

<F1>Plot Storm Centers, Movements, & Max Top
<F2>Range Markers
<F3>Erase Overlays
<F4>Filter
<F5>MESOs:0
<F6>TVSs:0
<ESC>Exit

F1 will plot the location of storm centers, movement in knots, and the maximum storm top in hundreds of feet (MSL). If the WSR-88D algorithms suggest hail is occurring with a given storm, that storm will be marked with a green triangle at the location where hail is suspected. Solid green triangles are used to denote storms that are considered producers of hail, while hollow green triangles denote storms that are probable producers of hail (Federal Meteorological Handbook No. 11). This option is automatically invoked after the precipitation is contoured. The top of the tallest storm (MT for max top in the RCM) is designated with a black square with white numbers and is specified just to the right of the location of the maximum echo top. To eliminate the wind barbs, maximum echo top, and hail from the screen, use F3.

F2 draws range markers.

F3 refreshes the screen with just the reflectivity pattern.

F4 will prompt the user for a level of precipitation to mask out. This can be a number from 0 to 6. 1 through 6 will filter out the blue, green, yellow, brown, red, and magenta colors, respectively. 0 will restore the screen to its original colors. In addition, precipitation can be masked out with the mouse cursor positioned on a desired color of reflectivity and by pressing the left button. It is easier to point and click the mouse cursor at one of the intensity squares at the bottom of the screen to filter out intensities.

F5 will display centers of 2-D uncorrelated shear, 3-D shear, and mesocyclones with 2 concentric yellow circles at each center.

F6 will display Tornado Vortex Signatures with red triangles.

<ESC> returns you to the Main Menu.

In addition, the program can produce several products from the wind data
including:

1) A hodograph, if more than 3 levels of wind are available. The hodograph shows what the cell motion (degrees/knots) would be based on the mean winds from the surface to 25,000 feet. The storm motion (in knots) is indicated for a storm moving 30 degrees to the right and 75% of the mean wind. In addition, the storm relative inflow vectors are indicated, with each color coded to match the winds in the VAD profile (Jarboe, 1993).

2) Calculations of storm relative helicity in m^?/s^? up to 3 km at 1 km intervals. The helicity values are computed for a right-moving supercell thunderstorm with a motion vector 30 degrees to the right of, and with 75 percent of the magnitude of the mean wind vector. This motion vector is consistent with that used at the National Severe Storms Forecast Center (Leftwich, 1990) and by the SHARP Workstation (Hart & Korotky, 1990). The storm relative helicity values are indicated in the hodograph square at the lower right hand corner of the screen.

3) Moving the mouse cursor to a particular storm with a motion vector (wind barb) will cause the calculation of a new helicity based on the storm's actual motion. Please note, when you move the cursor to a storm, the wind barb associated with that storm will change to red, and the helicity in the vicinity of that storm will be highlighted in the hodograph box with a yellow cross indicating the storm's motion.

4) The winds in the VAD profile that may be more representative of a distant storm may be changed. This is done by moving the cursor into the VAD profiler box to any particular wind that you want to change. You will observe, by moving the cursor through the winds, that colored circles representing the individual wind vectors will appear on the hodograph. Because not all winds below 10,000 ft are displayed in the vertical wind profile, additional colored circles will appear in the hodograph at intermediate levels of the wind profile to which the mouse cursor is pointing. When you find the wind you want to change, press the left mouse button. The mouse cursor will move into the hodograph box. Modify the wind to whatever you want. As you are changing the wind, the wind barb in the VAD profiler box will change accordingly, and the direction and speed label of the wind will also change in the hodograph box. When you are satisfied with your change, press the right mouse button and the mouse cursor will move back into the VAD profiler box to the point where you left off. Other winds in the vertical wind profile can be changed in this manner.

5) Moving the cursor into the hodograph box will update the calculations of helicity based on the position of the cursor (yellow + sign). The position

of the cursor is converted into a polar coordinate and is assumed to be the motion of a storm. For example, a northeast moving storm (240 degrees 30 knots) would have the cursor positioned in the upper right corner of the hodograph on the 6th ring. (Each ring is plotted at 5 knot intervals.)

Pressing <ESC> returns you to the Main Menu.

Notes on capabilities and resolution
There are a few things of which the user should be aware.

1) The RCM reflectivities are based on the 2.2 nm resolution composite reflectivity product. This means that reflectivities on the screen may or may not represent rain reaching the ground, since multiple elevation scans are combined to produce the RCM product. The image displayed is not necessarily occurring at the lowest elevation scan of 0.5 degrees. In many cases with severe thunderstorms, you will see a large area of level 1 reflectivity extending downwind of the strongest thunderstorm cells. This is indicative of the widespread cirrus cloud blowoff that is produced from these storms. Also, composite reflectivity products tend to make thunderstorms appear more severe than they may really be.

2) The RCM product is a 5.4 nm resolution product and echo "blooming" does occur. That is, spotty showers with cores only a few miles across will appear larger than they really are.

3) Two reflectivity display modes are available with WSR-88D, Clear Air Mode and Precipitation Mode. This radar program will indicate whether the RCM data was obtained in either the Clear Air Mode or the Precipitation Mode.

4) Anomalous Propagation/Ground Clutter does occur with radars and may not always be suppressed successfully. The worst of this phenomena generally occurs at night and in the early morning.

Program Information
Program Information and Procedures for Installation and Execution.

RCM DISPLAY SOFTWARE for IBM-compatibles

PROGRAM NAME: GDRCM

PURPOSE: Decodes/Displays/Manages WSR-88D reflectivity and wind data within RCM files. Includes an interactive hodograph, looping, and ground clutter suppression capabilities.

PROGRAM INFORMATION:
Development & Maintenance programmer:
Larry J. Hinson
Location: NWSFO Tulsa, OK

Languages: Microsoft Quick Basic 4.5, Quick C 2.0, Borland Turbo C++ 3.0 Original Release/Rev 1.6 - December 10, 1994

Running Time: Less than 30 seconds on a 486 DX2 to greater than 20 minutes on an 8088 XT. An 8088 XT with a co-processor will be significantly faster with an estimated running time of 3 to 5 minutes.

SYSTEM REQUIREMENTS:
IBM-compatible with DOS 3.3 or better, 512K+ RAM, 1.5 MB free disk space. EGA graphics capability monitor with 256K video memory. Co- processor and Microsoft-compatible mouse are recommended.

[Note: We are working on making this program available as part of the TAPR software library.]

Conclusion

In closing, I would like to add a few comments on data compression. When the RCM product and program appeared, the amateurs in the Tulsa area had been experimenting with ways to send pictures from the ATV system to the hams in Arkansas and other outlying areas who wanted access to the displays. We looked into ATV relays and packet. Both have their apparent problems. ATV links would cost a lot of money and the best we could do with compression of the ATV images was 50K using JPEG compression. These files would take from 15 to 30 minutes to send. Of course, the captured, compressed, and transmitted image was a raw reflectivity image and had no embedded wind data or storm and cyclone analysis. Thus the power of the RCM format became apparent. All of what we wanted and so much more could be delivered within a 2K to 4 K file and allow for multiple distribution.

While the approximate 6 mile resolution of the RCM sounds a little crude in some ways, it is usable, and I suspect that what we were digitizing on a field, or half-frame basis, with received noise, was no better, and perhaps would have been much older information when ultimately delivered. There is a lot of information packed into the small RCM data file and it appears to be used very well, and with the hardware that was already in place around the network. In some ways, the RCM files can be thought of as compression. It filters what is meteorologically significant, rather than trying to reduce just an image to a shorter form. A lot of the analysis is already done by the radar support system itself for the meteorologist.

The combination of the RCM message distribution across the network and the development of the GDRCM program has brought near real-time updates to EOCs and spotters of information that is required to fulfill their duties as amateur weather spotters. Combine the RCM capability with the weather alert functions and the network provides essential services to the day-to-day operations of weather related activities.

References:

1. Hinson, Larry. GDRCM. NWSFO Tulsa, OK. 1994.

2. McDermott, Tom, N5EG. Overview of the TEXNET datagram protocol. Proceedings of the 6th ARRL Computer Networking Conference. Redondo Beach, California. August, 1987. ARRL: Newington, CT. p.115.

3. Jones, Greg, WD5IVD and Tom McDermott, N5EG. Texas Packet Radio Society Projects: An Update. Proceedings of the 9th ARRL Digital Communications Conference. London, Ontario Canada. September, 1990. ARRL: Newington, CT. p. 122.

4. Morgan, Bob, WB5AOH, and Greg Jones, WD5IVD. An Update on TexNet and the Texas Packet Radio Society. Proceedings of the 14th ARRL Digital Communications Conference. Arlington, Texas. September, 1995. ARRL: Newington, CT.

An Update on TexNet and the Texas Packet Radio Society

Bob Morgan, WB5AOH (morganb@tenet.edu)
Greg Jones, WD5IVD (wd5ivd@tapr.org)

Texas Packet Radio Society • P. O. Box 50238 • Denton, Texas • 76206-0238

Abstract

This article shows the current status and updates the progress and accomplishments since the last published article in 1990 [1], of Texas Packet Radio Society, TexNet network, and other projects. The topics for this update cover the growth of the organization, the expansion of the network, the reliability aspects of the network, the latest firmware, and continuing projects.

Texas Packet Radio Society

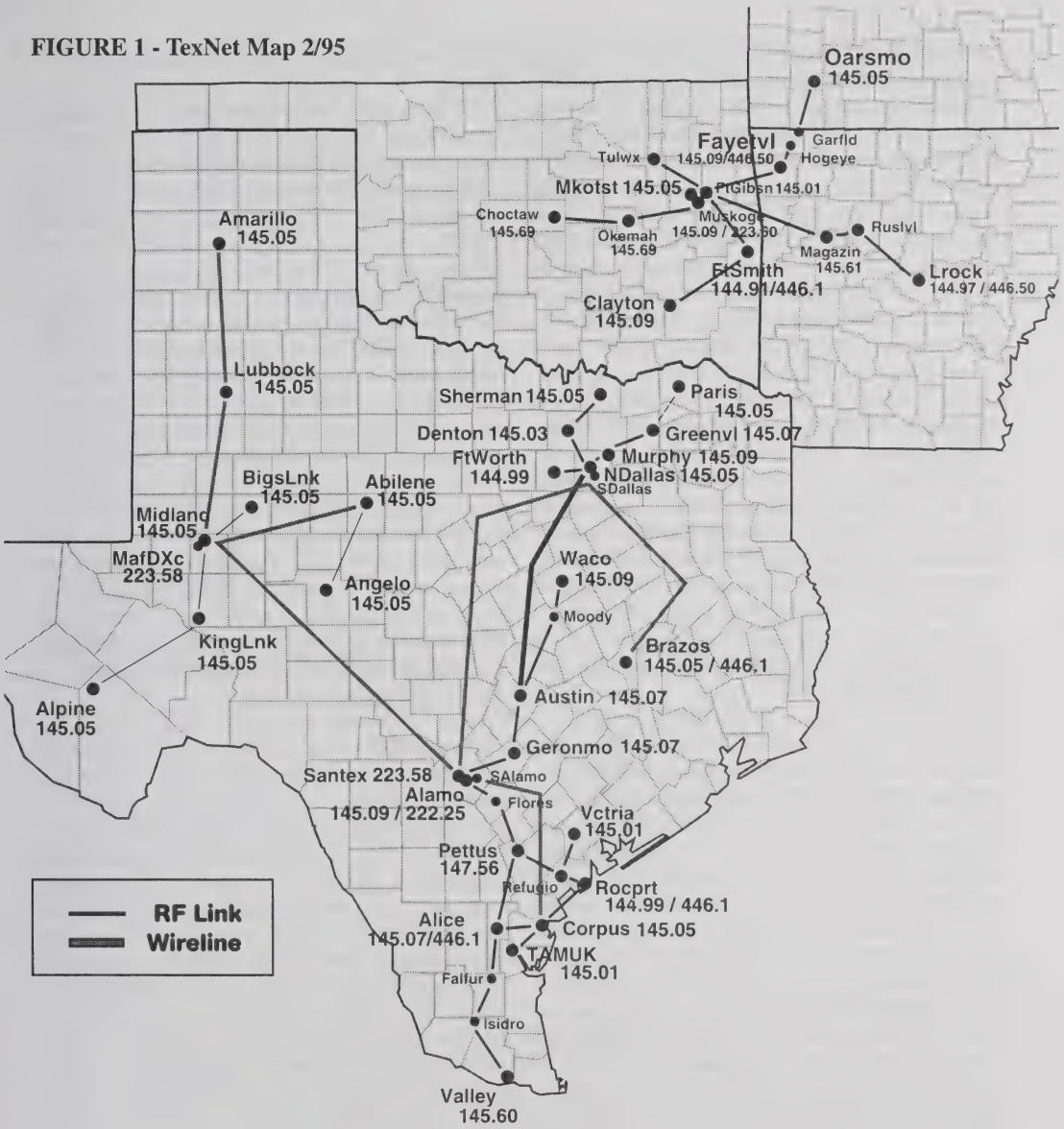
TPRS was founded in 1985, and incorporated in 1986 as an educational, public service, and scientific research non-profit corporation. The Texas Packet Radio Society's goals are: 1) design and research amateur radio packet networks, 2) provide education in the area of general packet usage, and 3) provide an emergency communication service network. The organization has members throughout Texas, many states, and several foreign countries and has had good growth since 1990. Membership in TPRS is now above 500 members.

TPRS itself does not support individual nodes per se, except for a few exceptions, like NWS, which serve the whole network. Individual nodes are the responsibility of local support. TPRS does arrange for support of certain projects (such as donated wire line or facilities) which are used to tie critical parts of the network together. This approach has been very successful in network management. In addition, the level of expertise required to build, install, and keep a network node running has provided a level of selection that has proved beneficial.

TexNet Network Expansion

Figure 1 shows the TexNet network map, which has now grown to include four states. The northern part of the network reaches into Missouri, in the vicinity of Aurora in the southwest part of the state. The network has also pushed all the way northeast to Little Rock, Arkansas, and northwest to the metro outskirts of Oklahoma City, at Choctaw. The southern boundary of the network is the Rio Grande River in the Texas lower valley. Cooperating organizations in adjoining states supporting this network include HogNet in Arkansas, OARS in Missouri, and WopNet in the Valley of Texas. Individual nodes are supported locally by individuals and organizations, and TPRS performs network administration, coordination, and management. Nodes continue to be added from time to time, and further expansions are being planned or are under construction. Approximately fifty full time service nodes are on the air at present, with a few more used for development, testing or construction.

FIGURE 1 - TexNet Map 2/95



Mileage spanned by the network trunks is phenomenal for an amateur packet network and breaks down as follows:

UHF RF amateur radio 9600b fsk	1559 miles	53%
VHF RF amateur radio 1200b afsk	147 miles	5%
Donated carrier wire line	1253 miles	42%
Network total trunk mileage	2959 miles	

The above mileages are point to point from site to site statute miles for active operating sites, as of 1995. In the case of the donated wire line circuits, city center highway mileages were used, and the various carriers deviate from these paths considerably, making the actual circuit mileages higher by some uncalculated amount. The network includes metropolitan coverage into three state capital cities — Austin, Texas, Little Rock, Arkansas, and Oklahoma City, Oklahoma.

End-to-end turnaround time from opposite ends of the network (800 miles), during times of relative network inactivity and smooth operation, are on the order of 15 to 30 seconds. Anything in excess of about 90 seconds indicates a problem in network operations. The network is not immune to problems of various kinds. Tom McDermott, N5EG, has gone into detail about the reliability of hop-by-hop networking with networks of smaller size and fewer nodes [2].

The user base of TexNet is composed of BBS forwarding, DX Cluster interconnections, real user-to-user keyboard contacts (some of which may use a network conference bridge), message servers for network users, Skywarn spotters and local EOCs, and network experimenters. The higher speed trunking (9600 baud) allows actual keyboard user contacts over wide distances. One of the reasons for the long standing decline and disuse of packet for keyboard contacts are the long response times typically observed on packet across the nation. I have personally participated in keyboard contacts spanning 500 miles or more on TexNet which were conversational and not painfully slow.

Network Challenges and Reliability

The usual challenges and setbacks in operating a network are the month-to-month repair of occasional failures of nodes due to catastrophic events such as lightning damage, or moving of nodes from one site to another in search of more amiable "landlords." TexNet has had problems keeping a site active in order to extend the network between Oklahoma and Texas across the Red River. For over the last year, the north and south parts of the network have had to operate independently until a new network site can be located and brought on-line. Sometimes, we lose a site here and there, and have to regroup and look elsewhere. Meanwhile, the network is broken in the middle. So has been our luck over the years at the small minority of sites where we have had these problems. Overall, we have enjoyed a good record of keeping all of the sites intact, and some lesser degree of luck at keeping lightning and surges out of the equipment.

A major disturbance to network operations that occurs from time to time in South Texas (San Antonio, Texas south) is the operations of airborne radar. The 70cm amateur band is a secondary allocation for amateurs with the primary allocation assigned to government radiolocation. In this case, this takes the form of the airborne radar being used in the last few years to intercept drug smuggling in the Gulf of Mexico. From time-to-time on an intermittent basis, this portion of the network is interrupted by radar interference. Most of the time the network stays intact, and only a minority of the time is the network affected.

Another factor in network performance is maintaining the trunk radios on frequency. Network operators have done a good job of putting 9600 baud FSK into the field and making it work well, which it has most of the time, since 1984. The one detractor from that, which has earned the network a

"fragile" reputation, is keeping the UHF radios inside a very tight frequency tolerance. Network node owners have to keep their radios on frequency with a lot of sites and few site visits to spare each year. Thus we expect the network to hold up through both summer heat and winter cold at some sites.

We deal with a signal that is almost too tight for the I.F. filters in the radios to pass. It does fit just fine if the radio stays within a few hundred cycles of where it is supposed to be, but this is a tall order at UHF and requires tighter tolerance than the equipment was specified for when it was designed for two-way voice service. Throughout all of our design and construction we have resisted the urge to remove I.F. filters, as some 9600baud conversions have done, because of the introduction noise. We know from experience that the network runs very well if the frequency is kept under control.

Yes, we DC couple, all the way from the modulator, through the transmitter and the channel and the receiver, through the modem to the slicer, for a better error rate. This does place more stress on the frequency control. As long as the node can hold frequency, the result is a better error rate at the slicer. Note, that in contrast to some other 9600baud packet methods in use today, TexNet uses pseudorandom scrambled NRZ, as opposed to NRZI, for tighter bandwidth, which does allow the network to achieve excellent performance with existing narrow filters with the commercial 5 KHz voice deviation two-way commercial land-mobile radio equipment being used. Deviation for data is set to approximately 3 KHz.

We have found it necessary to "age" brand new crystals which we receive from reputable suppliers. This was never much of a concern with the same equipment and suppliers in voice service. There has been some debate concerning whether the manufacturing practices of the crystal industry have changed in the past few years, possibly due to such things as solvents being discontinued and the like. New crystals are operated at elevated temperatures for a few days or weeks prior to installing them, and even then we see drift in the first year of operation, usually requiring readjustment at varying intervals.

In an effort to find a better solution to frequency control over the winter season change, it was found a one component temperature regulator (the positive coefficient thermistor) would work when attached to the crystal can or the channel element. When fed with 12 volts DC the thermistor holds the element within a few degrees of approximately 30 Deg. C. Most of the main trunk radios in Oklahoma now have them and have shown good results.

Network Radios and Modems

Over the past few years, our technical developers have experimented with various surplus two-way radio makes and models to see which are most suited for use in 9600b FSK service in the remote site environment we frequently encounter. The original surplus radio of choice was the RCA model 700. Since then, we have tried several Motorola, GE, and Johnson models. As a general rule, we have found a true discriminator to be superior to anything using a quadrature detector, and we have found some quadrature detectors to be better than others. The radio of choice in the Oklahoma environment has become the Motorola Micor. The Motorola Mocom-70 has seen some use also. The Johnson 6060 has seen some use in Texas. We have had some reliability problems with the venerable RCA-700, due to the failure of some PA components for which replacements are not available. Some of the newer model radios we have tested have a coupling capacitor downstream of the detector, and we find these models to be problematic with end-to-end DC coupled network.

We use one, and only one, RC time constant in the entire FSK signal path, and it is in the slicer of the modem receiver, and its time constant is optimized for the time required to acquire the received packet's DC center. Any other unintended series RC circuit along the way contributes to baseline "wander," and ultimately to receive errors. Any RC network which is part of an original voice radio design is almost certainly NOT optimized for the proper time constant anyway. Very recently, Tom McDermott, N5EG, has developed and is testing a newer improved slicer, which is a lot faster and more accurate than the one we now have. It seems to be a promising development. If successful, it will take the form of a daughterboard for the existing TPRS modem. It locates both extremes of the received signal eye pattern, and sets itself midway a lot faster than the older, simpler RC averaging comparator. It might allow the network to operate with a shorter TxDelay timer.

Network Code Continuing Development and Revisions

Version 1.6x was just coming into use at the last CNC report in 1990 [1]. This version was our standard firmware on the network from 1990 through 1993, and was a pretty well behaved product. This version of the code showed excellent performance and reliability. Previous versions were subject to dropping into a state of disconnection when the network was idle. Currently things stay connected as long as the path is there. Unfortunately, it works too well at times. During times of abnormally long range UHF propagation (up to 500 miles in one hop), the network forms rogue routes which do the network no good, and of course, don't last. They certainly disrupt the shortest path algorithm and leave the routing tables in disarray. This phenomenon continues to the present, but we are on the verge of solving most of this problem with the next version (v1.72), currently in distribution.

Version 1.70 was introduced in 1993. Primarily it is version 1.6x, with the routing tables expanded from 50 nodes to 90, and a few cosmetic changes. Version 1.71 was almost the same, but had a minor bug fixed with the DWAIT timer, and allowed for improved performance on shared channels, such as a multidropped wire line circuit we use in West Texas. It has all of the characteristics of the 1.6x versions when it comes to adaptive routing and the corruption of its tables when the band opens. Another change is that the NCP and TNC-2 versions of the code are now handled as a pair of similar products, and are updated together. This overcame a time lag we had prior to that in distributing the TexLink (TNC-2) revision when revisions were made to the NCP version.

Version 1.72 has just now left the testing stage and is being installed throughout the network. It is one of the biggest revisions to TexNet in years. It will soon be in distribution through TPRS and TAPR libraries. Here is an overview of what this software does.

1. TexNode. There is a brand new TexNode service. This is a local node, which is a connected alternative to digipeating. It fills in a gap in TexNet which allows stations which can both hear and work TexNet on the same node (not necessarily on the same physical port or frequency) to work each other without digipeating through the node. The TexNode is accessed by connecting to the node's -1 SSID, which is new. The TexNode is still new and experimental. It may have some problems with flow control. Experience and use will show what its shortcomings are and how to solve them. It will probably need to be fixed at the next opportunity for a revision.

The TexNode provides three pairs of connects for locals to use to connect to each other, without the drawbacks of digipeating. Please note that digipeating was originally invented in the early days of AX.25 as a

temporary stopgap to produce some extended range connectivity while waiting for layer 3 networks to appear, which of course, they have now. Digipeating has drawbacks which have been discussed many times, and has been discouraged or locked out in some places [3].

There are a few nodes on TexNet which do not support digipeating for local QSOs, and as this local node service is expanded (and any bugs fixed) there will probably be more digipeating discouraged or locked out.

2. Weather Alert. There is a weather alerting service which is allied to the Weather PMS. A PMS, or packet message server, is a message server integrated within TexNet, and will be explained later. The Tulsa PMS is now generating a UI broadcast on most of the NE Oklahoma TexNet ports every time a type "S" or severe weather message is saved to disk from the National Weather Service (NWS) input.

The UI that is sent on all ports of the network includes a bell, and includes the heading, and the first few lines of the message — enough to determine the counties of interest. It is sent from an alias callsign of "ARES", and anyone wanting to monitor these can put ARES into their Budlist.

These broadcasts can also be initiated manually by a station with sysop privileges, such as a net controller to announce a net activation. The network can be internally partitioned to limit the geographical distribution of these broadcasts. Presently the only partition is at the Red River into Oklahoma.

There has been some addition to the Weather PMS input handling, and it is now feasible with standard code to upload the raw weather source from one site to another, where a PMS can be located remotely from the source, or to relay input data from one PMS to another. When we consider sites for weather PMS input, this feature needs to be considered. It is possible now to use a TexLink node (the TNC-2 with a TexNet EPROM) to act as the uploading input node, whether or not it has a disk drive. Of course, the standard NCP or possibly the NCP-PC may be used. Presently, there is some fairly widespread investigation behind the scenes to acquire more weather input sources, using the highly successful TULWX Tulsa, Oklahoma site as a model [4]. The Texas network went through spring storm season without a Texas weather PMS, as we lost that site at a critical time.

3. New Internal Datagrams. There are a few new internal datagrams to support new functions within the network, and one new network information code, NIC-22, which users may encounter from time to time during routing errors.

This particular error comes about as a result of a datagram propagating through the network and encountering a node which has no knowledge of the return routing for that datagram. It now generates an error which returns on the same channel it was received on, directed back at the source, while the return channel is still known. Previously, there was no check, and a deadlocked condition existed when the response could go no farther on the return trip. In addition to the error completing the "response," the receipt of the error automatically issues a routing correction to the node finding the error to begin with, so that subsequent uses of that route will be successful.

This release adds some major internal modifications and tuning to the basic routing. The routing methods themselves are unchanged, but it is much easier for the routing manager (i.e. Harry) to maintain the system. He now has the capability to edit any route table, whereas he used to either use a sledgehammer approach, or execute cumbersome memory changes. The release also addresses the root cause of many of the

routing corruptions, the unwanted trunk paths during band openings, by limiting the formation of such paths in the first place. When 1.72 is fully installed over the network, expect that there will be fewer corruptions. The ones that do occur will be much easier for Harry to fix, so the network should run better more of the time with less of his attention. This software also implements a remote function needed by the automated NetMgr (a Cardnal function) to perform automated routing management in the future.

4. Remote User Listing. There is now a Remote Users listing command. It displays the stations connected to a remote node (presently it does not support the local ncp), and displays some numeric information which can be decoded to reveal which L3 service the user, or trunk, is on, what the L2 state is, and some allied information. It displays both users and other connected trunks. It is useful to both users in general to find out who is on any node and to network sysops to determine the status of trunks.

5. Improved Local Console. For those node operators who have terminals installed on the local port of the node, the terminal port has changed from 7E1 to 8N1, and some bad behaviors of the Console program have been fixed. The input line will no longer force back to command mode if too long a line is typed. PTRACE has been extended a little bit to include hex dumps of some of the datagrams, recognition of NetMgr datagrams, and a bad bug fixed which locked up the works from time to time. For those operators who have their node implemented on an NCP-PC card, the console port may be operated over the bus instead of through the serial port, via a supplied INT14 TSR.

6. New Stats Display. Information has been added to the node statistics display and the format is completely different, and does allow for it to fit on a screen. However, an unfortunate result occurred and the date now appears in an insane format. Due to code space problems, I'll apologize beforehand for the unfortunate result. With the exception of the screwy looking time/date, it is a lot easier to read. Added information includes TX frames aborted, the revision level of the software of the node, the CPU effective speed raw data, a ROM checksum and the dates/times of initial powerup and subsequent firecode and warm boot restarts of the node. One good result is that no longer does one have to multiply some numbers by 10; they are displayed with a trailing zero so you only have to read the number. Another good result is that the amount of program memory occupied by the statistics routine was cut in half and left room for some of the new features of this release.

7. Network Path Locking. The algorithm implemented in v1.72 follows a general trend that has been seen in other packet networks of route locking, or more accurately called PATH locking. The routing tables in each node are available at all times for updates by the shortest path algorithm. What is locked out is the formation of temporary shorter paths at layer 2. This would seem to be at odds with a system developed over the years for automatic addition of new nodes to the network. To allow for the addition of new nodes (or old nodes out of service for an extended time), there is a way around it — the 15 minute timer. Since v1.6x, all nodes broadcast their existence to the world every 8 minutes, inviting any TexNet node hearing the broadcast to connect, new nodes or old nodes, close or distant. An exclusion list of up to 5 nodes is sent along with the invite. If the two nodes are not on the exclusion list, v1.6x allowed the connect to be made, whether it was 500 feet or 500 miles. Whether the nodes were intended neighbors who had briefly lost connection, or distant and unrelated to each other, was irrelevant. This scheme has done a good job of maintaining the integrity of the connections

we wanted, but also opened up a lot of transient rogue paths during band openings, often short-circuiting the distance counts in the tables by 5 or 10 hops, which destroyed the intended stable routing. Version 1.72 now excludes new paths which are not shown as direct neighbors in existing routing. The exclusion is inactive for 15 minutes after a node is reset and started up. The 15 minute timer may also be started and stopped remotely by an existing manual remote control command. This is one of the means by which a new node can be added to the network. The other means is to manually insert the required entry in the table.

Network Code Testing and Future Directions

This software was extensively tested in ALAMO, one of the highest traffic nodes on the system, for a couple of months prior to releasing it. Thanks to Harry Ridenour, N0CCW, and Clarke Diekmann, K3WGF, and all of the local and remote users of ALAMO who got bumped off in the process of debugging, for seeing the process through. ALAMO was chosen because it is easily accessible to swap eproms, in addition to being a high traffic node, and also to give Harry the earliest access to the new route editing functions. It was also tested on TULWX and neighboring nodes in NE Oklahoma. As of July 8, it was installed in ALAMO, SANTEX, SALAMO, FLORES, ALICE, WACO, TULWX, LROCK, MAGAZIN, FTSMITH, CLAYTON, FTGIBSN, MUSKOGE, MKOTST, DENTON, SHERMAN, MURPHY, GREENVL, AUSTIN, AUSDXC, and GERONMO, and is ready for a few more. I have e-mailed a copy of it to Jay Nugent for GL-NET to start using.

This software is available in both NCP and TNC-2 versions, as were 1.70 and 1.71. In addition, it is easily adaptable to creating a specialized weather PMS, such as TULWX, without many changes other than preparing the weather product database. Previously, a weather PMS became almost a separate programming task, and this is less true now. As in 1.70 and 1.71, the software supports 90 nodes per network. I estimate that I spent 300 to 400 manhours on this release and it was a major undertaking. Some of the route management improvements are items I have been thinking about for the past four years or so, as I have watched the network routing get corrupted myself, and listened to Harry's observations also. Most of the weather PMS improvements came out of a meeting held at the Green Country Hamfest in 1994 and some more came from Steve Piltz and Brad Smith in Tulsa.

I have plans for a subsequent version. Some of the revisions I would like to add would require that all Eproms on the network prior to 1.72 be replaced with 1.72, since I foresee some incompatibilities being made. My biggest reason to want to get 1.72 installed quickly is to solve our routing corruption problems, and those sites experiencing unnecessary trunk connects during openings are the prime priority. The downside to future code development is that about 30 bytes of space is left in the NCP Eprom, so there isn't a lot to work with. Some space will come from shrinking existing programs by either eliminating lesser used functions, or from time consuming recoding to make the code shorter. Some of the new features have actually been implemented by programming tricks which resulted in shorter code. There are also efforts underway to integrate and/or supplant this code with a TCP/IP interface to extend the range and the lifetime of the existing and growing network.

Network Code Development

The code itself is written in multiple linked modules of Z80 assembly language, using a native CP/M assembler/linker running as a CP/M program under Z80MU, which is a Z80 emulator and CP/M emulator running

under DOS on a PC. A complete assembly and link process can be performed in about 20 minutes on a fairly fast 386 system, producing a generic binary load of either the NCP or the TNC-2 code. We then use an editing process to perform a binary edit to customize an image for any particular node, to add its callsign, net ID, parameters and so forth, and all of that is burned into an eprom. Except for abnormal operating conditions, no further entering of parameters is required after the node is started up.

The NCP-PC and NetMgr Projects

The NCP-PC project continues, although no more kits are planned. The NCP-PC is an I/O card for the PC, which includes the cpu section, less modems, of the TexNet Z80 NCP, and behaves as a very loosely coupled coprocessor. It behaves as an EPROM to the Z80, since writes are inhibited from the Z80 for that portion of the memory map. It is entirely RAM and can be written by the PC at random. It makes a fine development system, since the step of EPROM burning is not needed. All of versions of 1.6x and 1.7x were developed using a NCP-PC. While it doesn't emulate a TNC-2, the only differences between the TexNet NCP and TNC-2 code are the number of serial ports, differing addresses, differences in the memory boundary between EPROM/RAM, and the presence of a CTC chip on the NCP or NCP-PC. These differences are allowed for in the conditional assembly of the standard software suite for both products. When acceptable code for the NCP is achieved, the conditional assembly is rerun for the TNC-2 product, and burned into an EPROM and tested conventionally in a TNC-2, usually with no surprises. The TNC-2 code has in fact been released in finished form under 1.70, 1.71 and 1.72 without further modification.

The automated NetMgr (Network Manager), a NCP-PC project kept on line co-located at the Murphy node (the town of Murphy, Texas, not the errant destroyer of mankind's engineering achievements by the same name) continuously monitors and logs network activity. This information is then analyzed to determine potential network problems. There has been slow progress made on the automated route management R&D phase of this project. The release of v1.72 added a needed inquiry function to help facilitate that project. The eventual goal of the project is to have all automated route management handled by the Network Management system. The statistics it generates enables better manual routing maintenance to be performed, than would be possible were the statistics not available. In defense of the goals of automated NetMgr, it should be noted that the energies of the developer (Tom McDermott, N5EG) were diverted to not one, but two DSP projects. The first being the TPRS DSP land-line modem and the second one being the TAPR/AMSAT DSP-93 project. The NetMgr is still functioning at v1.6x and there is some programming effort planned in the near future to bring this one node up to current. That is expected to be a minor job, but crucial to NetMgr's operation.

Network Administration

The network still relies on human beings to perform network routing management manually. This job is being made easier under v1.72 with a remote route editing command. This manual command makes use of some logic added in v1.6x to allow for the automated NetMgr to add and change routing remotely. At that time, we had no manual means to do so, other than a remote memory patch, or to selectively announce or reset nodes to reinitialize the autorouting. This follows the general trend of all automation, that the human is still sometimes not replaceable, but some of the

drudgery can be automated, leaving the human to take care of the exceptions. Goals in v1.72 were to reduce some of the extraneous inputs which disrupt the routing in the first place, to give the human manager the editing command he needed, and to add the missing inquiry function that the automated NetMgr needed. Eventually the NetMgr can take over some jobs from the human in the future. It should be noted that we are blessed with a human net manager, Harry Ridenour, N0CCW, who spends some time every day watching over the network and seeing to its integrity. A lot of the things he does are not of the nature that can be automated. These include coordinating with site owners for site access, wire line troubleshooting, arranging new site installations, and handing out new node numbers. The new network remote function allows the network administrtor to easily change network routing information from anywhere on the network, thus creating more time to devote to other tasks.

TexNet enjoys an environment of administered planning. Our administrator governs the distribution of node numbers to sponsors who fit into an overall plan for a network. It is not a network that expands and contracts on a cyclic or a whimsical basis. We know in advance when a node will be added or repaired, and can make the necessary commands to the system, to either preinstall the proper entry in a routing table, or start the timers at the proper time. We have learned very well not to bring up or reset new nodes during band openings, and if we have to, how to get the route tables into a proper state after we have done so. Version 1.72 will make this job a lot easier. We anticipate development of some automated NetMgr algorithms and eventual auto routing control.

Other services provided by the CARDNAL node include a multiuser file server, an amateur callsign lookup database based on a CD-ROM callbook, an ARRL bulletin capture system based on reception of the AMTOR-FEC bulletins (retaining the most complete recent copy), and a network wide conference bridge. The file server is also the delivery vehicle for the ARRL bulletins and the NetMgr statistics files, in addition to other files loaded to the system by the users or the host.

Other TPRS Projects

TexNet has several donated land line circuits that make up some of its routes, although TexNet is primarily an amateur RF based network. We have access to 200+ miles of point to point circuits which are used, from various land line commercial providers, for nothing but our good will, a smile, and probably our tax exemption number.

Land-line circuits are an integral part of the network, since ninety miles on a calm damp evening is the best intermittent RF circuit we can boast of, or 65 miles if you want a "reliable" RF circuit. Without these long-haul paths in the network, several sections would be unattainable as network paths (i.e. trying to find enough amateurs between Austin/Dallas/San Antonio and the West Texas cities to support nodes).

Surplus GDC 209 synchronous modems are used to drive these land line circuits. Due to reliability problems with the GDC modems, a project was started to design a modem that would give us the necessary interface as well as provide needed monitoring and testing functions. Tom McDermott, N5EG, set about to solve two problems with this modem with a replacement based on current DSP techniques. The first goal was to put a reliable modem into service. The second goal was to speed up the circuits we were using them on, because the 209 standard has an abysmally long training sequence which it must perform on each transmission of a

multidropped circuit. While it is 9600baud, the txdelay required to support the modem training sequence is about 400 mS. In most cases, this is longer than the packet itself, and certainly so for the acks and supervisory packets. These circuits have a real throughput problem even when the modems are in good shape. Tom has designed and tested a prototype replacement which will emulate the constellation of the 209, and can either emulate the training sequence, or use an abbreviated training sequence which is based on recognition of previous recent transmissions of the various modems operating together on the circuit. The design includes remotely commanded diagnostics and remote level adjusting software, designed to work on an entire circuit of modems from only one end. These remote diagnostics features allow one network manager at a site to test all land-line modems without requiring operators on both ends of the circuit. Oftentimes on our freebie "leased" circuits, we never have the luxury of a technician on each end of a circuit under maintenance, and this software should solve that problem. In the past, trying to arrange access to sites at the same time for end-to-end testing has proved an impossible task.

Current status of the new modem is that boards and parts have been gathered and are awaiting to be built-up for installation in the next few months. With luck, this new device will improve performance and reliability on our land-line circuits a good deal.

Conclusion

TPRS and TexNet are alive and well and continue to develop and expand. We hope to be able to update their status many more times in this venue in the years to come. The success of the design of TexNet can be attributed to: higher speed trunks, separated user and trunk channels, low network-ip overhead, low installation cost per port, tight trunk bandwidth with low error rates, network planning/administration, and multiple applications available via the network to the user population.

—

Reference:

1. Jones, Greg, WD5IVD, and Tom McDermott, N5EG. Texas Packet Radio Society Projects: An Update. Proceedings of the 9th ARRL Computer Networking Conference. London, Ontario Canada. September, 1990. ARRL: Newington, CT. p. 122.

2. McDermott, Tom, N5EG. A Primer on Reliability as Applied to Amateur Radio Packet Networks. Proceedings of the 13th ARRL Digital Communications Conference. Bloomington, Minnesota. August, 1994. ARRL: Newington, CT. p. 122.

3. Ridenour, Harry, N0CCW. Digipeating. Packet Radio: What? Why? How? TAPR: Tucson, AZ. 1995. p 61.

4. Morgan, Bob, WB5AOH, and Greg Jones, WD5IVD. The Tulsa National Weather Service TexNet Interface Project. Proceedings of the 14th ARRL Digital Communications Conference. Arlington, Texas. September, 1995. ARRL: Newington, CT.

DSP-93 Programming Hints

Frank H. Perkins, Jr. WB5IPM (fperkins@onramp.net)

Introduction

The DSP-93 is surprising user-friendly to program, considering it has a 40 MHz Harvard-architecture DSP processor under the hood. Applications can be successfully developed with only a PC and an oscilloscope. So far, more than a half dozen radio amateurs have developed and published applications for the DSP-93, including N5EG's windows-based oscilloscope and spectrum analyzer displays, W3HCF's super-hot HF modem, a collection of satellite and terrestrial modems by this author, plus Mac versions of the spectrum analyzer and oscilloscope displays by W5RKN.

In this paper, I offer you several hints on programming the DSP-93 that will hopefully get you around a couple of rough spots I have encountered. These hints are intended for someone with a working knowledge of assembly language programming, the 320C25 instruction set and the Programming Guide for the DSP-93[1].

DMOV, MACD, LTD, etc.

DMOVs and instructions with embedded DMOVs only work in *internal* data memory. This is often a big surprise to programmers that are familiar with the 320C26 DSK kit, which only has internal data memory. The DSP-93 can have up to 64K of data memory. However, DMOV, MACD, LTD, etc. will not work properly in the external RAM segment of this memory. Be sure you keep your data delay lines for FIR filters, correlators, scramblers, etc. in the internal data memory segment!

Data to Program Memory Remapping

To support adaptive filters, the 320C25 can remap data memory at 0200h to 02FFh to program memory at 0FF00h to 0FFFFh. Avoid the temptation to put any tables or code in the 0FF00h to 0FFFFh segment of program memory, as it will *disappear* if the CNFP instruction is invoked.

UART Data Transfers

Remember that UART read/writes, which are done with 16 bit IN/OUT instructions to data memory, are only valid in the lower eight bits. Be sure to mask the upper eight bits to zero.

AND, ANDK, OR, ORK, XOR, XORK

ANDK, ORK, and XORK support mask shifting into the high word of the accumulator; AND, OR and XOR do not. AND zeros the high word of the accumulator. ANDK zeros all bits above and below the shifted mask and always zeros the MSB of the high word, regardless of the

shift. OR and XOR do not affect the high word of the accumulator. ORK and XORK do not affect bits above or below the shifted mask and do not affect the MSB of the accumulator, regardless of the shift. The somewhat subtle differences in high word treatment between memory addressable and immediate versions of these logical instructions can be confusing, especially if sign extension is set.

Nonlinear Operations

Nonlinear operations can be bandlimited or not bandlimited. Due to alias residues, nonlinear operations that are not bandlimited can create some real surprises. For example, in my first attempt at an APT demodulator I used the ABS operation to do a full-wave rectification of the 2400 Hz amplitude-modulated carrier signal, followed by a low-pass FIR filter. Bad idea. I had a 10-15% "hum" signal in the demodulated output.

After hours of circuit troubleshooting looking for the source of the hum, I started to realize I had created it in the DSP math. I did a simulation of the APT demodulator in QuickBasic and the hum was there! The problem was that taking the absolute value of a sampled analog signal was not a band-limited operation, and had created many, many harmonics. One of harmonics landed just above or below the sampling frequency and was aliased right into the passband of the low-pass filter. When I squared the signal to detect it, the problem went away. Squaring a signal is a band-limited nonlinear process, and only creates a second harmonic. If your sampling rate is at least four times the highest frequency in the incoming signal, alias residues will not be a problem for square-law detection or product detection, which are both band-limited nonlinear processes that create only second harmonic components.

Clipping, half-wave and full-wave rectification are examples of nonlinear operations that are not bandlimited. These operations can be used under certain circumstances, but be sure to do an alias-residue study before using them.

Analog and Digital SW Probes

My basic code debugging tools are analog and digital software probes. I try to dedicate pointer AR7 for the analog signal probe:

```
; Initialize the probe pointer in your initialization routine as follows.

PNTR_INI   LRLK      AR7,07Eh    ; point AR7 to 07Eh (unused location)

; Then place the following code just below an operation you want to check,
; and reassemble. This code will load the target variable, shifted as needed for
; scaling, into the probe buffer.

PROBE      LARP      AR7         ; make AR7 pointer
           LAC       XXX,Y       ; get variable, shift to scale
           SACL      *           ; store @ probe
```

```
; Then use this code near the end of the DSP routine to output the probe
; buffer to the D/A converter, where it can be viewed with an oscilloscope.

AIO_OUT   LDPK    00h       ; data mem page 0
          LARP    AR7       ; make AR7 pointer
          LAC     *         ; get probe
          ANDK    0FFFCh    ; mask out AIO control bits
          SACL    DXR       ; AIO out
```

For a digital probe, I move this code just below the logic operation I want to check:

```
; load the buffer containing the target bit and mask it off

TST_LGC   LAC     XXX       ; get buffer containing bit to output
          ANDK    YYY       ; mask off bit

          BZ      BIT_LO    ; if bit = 0 goto BIT_LO

BIT_HI    LAC     DO        ; else load DO
          ORK     04000h    ; OR RDI bit to 1
          SACL    DO        ; store DO

          B       LGC_OUT   ; goto LGC_OUT

BIT_LO    LAC     DO        ; else load DO
          ANDK    0BFFFh    ; AND RDI bit to 0
          SACL    DO        ; store DO

LGC_OUT   OUT     DO,06h    ; output to TNC port RDI bit
```

In this case, the probed data bit appears on the RDI output line of the DSP-93 modem disconnect header. On my unit, this line is very easy to reach with an oscilloscope probe. I often compare two logic signals by putting the second signal on the RCLKI output line.

AIO Low-Pass Filter Programming

The sampling rate Fs of the 32044 AIO chip in the DSP-93 is given as:

Fs = 5000 / (Ta * Tb), in kHz

However, the low-pass input and output filters in the AIO are programmed by the value of Ta only. The cut-off frequency Fc for these filters is approximately:

Fc = 60 / Ta, in kHz

Take the case of an 8.681 kHz sampling frequency. Ta * Tb = 576. Assuming we want a 2.5 kHz cut-off frequency:

Ta = 60 / 2.5 = 24, and Tb = 24

Had we chosen Ta = 12 and Tb = 48 to achieve the same sampling rate, Fc would now be 5 kHz, and input signal aliasing and poor output signal reconstruction could be problems. In picking Ta and Tb values, expect to do some juggling between the sampling rate and the cut-off frequency of the low-pass filters. Remember, for linear signal processing you want Fc less than 50% of Fs and for band-limited nonlinear processing (second-harmonic generation only) you want Fc less than 25% of Fs.

References

1. Parsons, D Haselwood, B. Stricklin, *Programming Guide for the DSP-93*, TAPR, http://www.tapr.org/tapr/html/dsp93.html, 1995.

2. TMS*320C2x User's Guide*, Texas Instruments, 1993.

NETMGR: A Graphical Configurator for ROSE X.25 Packet Switch Networks

William Slack, NX2P
Donald Rotolo, N2IRZ
Radio Amateur Telecommunications Society
PO Box 93, Park Ridge NJ 07656 USA

Abstract

NETMGR is a windows-based graphical configuration utility for ROSE X.25 Packet Switch networks. This paper describes the features and usage of the software in detail.

Introduction

NETMGR is a windows-based utility that provides an easy and intuitive method of building configuration files for ROSE X.25 Packet Switch networks. NETMGR allows the network manager to draw the network in a graphical fashion, enter the relevant data and then have NETMGR automatically build the initial configuration files for the switches. In many cases no manual editing of the configuration files will be needed.

The switchop simply draws the network using an interactive drag-and-drop system, and then enters the relevant data for each site and switch. In a single step, NETMGR builds the configuration files for the entire network as drawn. In most cases, no manual editing of the resulting configuration files is required.

Starting NETMGR

Once NETMGR is installed, simply double-click on the icon. Click OK at the startup screen, and the main screen, shown on the next page, will appear. To view a previously drawn map, select 'File' 'Open' and select the desired map file, which will appear on the screen. To draw a new map, follow the steps below. It is recommended that a new user view the sample map file to get an idea of how things function before drawing a new map.

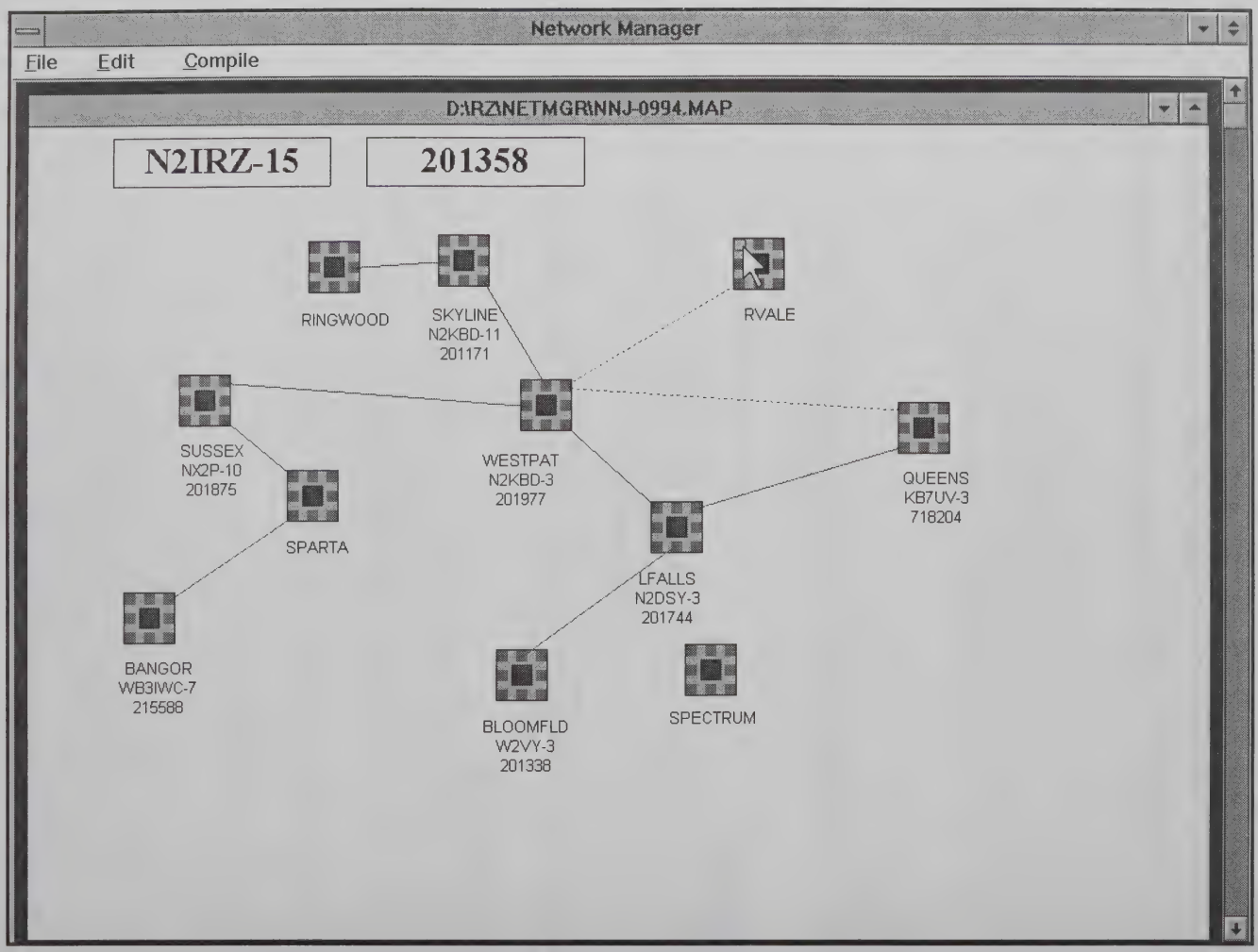
Drawing the Map

There are four steps to drawing a map of your network:

1. Place the sites
2. Define the site global information
3. Define the switch-specific informaton
4. Draw the links

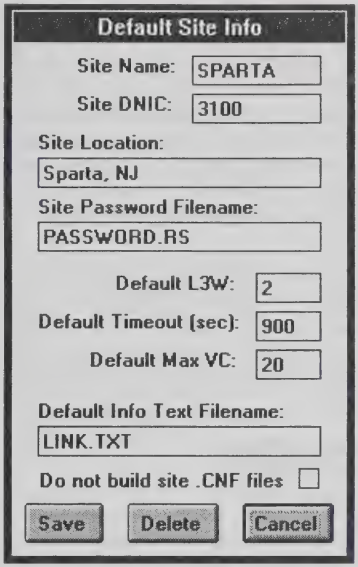
Placing the Sites

To start drawing a map select the "Add Site" option under the File menu. This will cause a new site icon to appear in the upper left corner of the map window. The site icon looks like a tic-tac-toe board: eight small squares surrounding a central square of equal size. The eight outer squares represent individual switches, while the center square represents the site itself. This icon can be moved to any spot on the screen by by using a simple left mouse drag/drop operation. To do this press the left mouse button while the mouse pointer is over the center of the site icon. Then while the mouse button is still down drag the site to its new location and release the mouse button.


Network Manager
File
Edit
Compile
D:\RZ\NETMGR\NNJ-0994.MAP
N2IRZ-15
201358
RINGWOOD
SKYLINE
N2KBD-11
201171
RVALE
SUSSEX
NX2P-10
201875
WESTPAT
N2KBD-3
201977
QUEENS
KB7UV-3
718204
SPARTA
LFALLS
N2DSY-3
201744
BANGOR
WB3IWC-7
215588
BLOOMFLD
W2VY-3
201338
SPECTRUM

Defining Site Information

Once the site has been placed next you will want to enter the default site information. To access this information, right click on the center of the site icon. This will bring up the default site info dialog that will look something like the following.



Site Name is a label up to 8 characters long that is used as a reference when building the configuration files. It is also used as the caption for the site icon. Do not place any spaces in this label or it will mess up the configuration files. Also used for the $N token in the info text.

Site DNIC is the DNIC (Data Network Identification Code or country code) for all the switches at the site. Also used for the $I token in the info text.

Site Location, usually the city in which the site is located, is a text note that can be referenced by using the $L token in a information text file. The $L token will be replaced with the text entered in this box thereby allowing a generalized information text file to be customized for each switch when NETMGR builds the configuration files.

Site Password Filename is the filename that contains the password for all the switches at this site. This filename is the same filename that you would put in the configuration file of a switch if it is to have a password. Leaving this field empty causes NETMGR to build the configuration files without passwords.

Default L3W is just what it says. To simplify the job of the network manager a default L3W can be set for the entire site which than can be overridden by individual switches if neccessary. L3W is the "Level 3 Window" which specifies the maximum number of frames for a single virtual circuit that can be transmitted in a single burst. MAXFAME defines the total number of frames that can be in a burst. Therefore setting L3W below the MAXFRAME value insures that a slow link does not hold up data on another link.

Default Max VC is the default maximum number of virtual circuits. As indicated above for L3W, each switch can have a MaxVC that overrides the site default, however entering a site default relieves the network manager from having to enter the same value for each switch at a site.

Default Info Text Filename is the site default information text filename. The information text filename is a standard ascii text file that NETMGR references which contains the info text that will be used in the configuration file. The site default text acts in the same way as default L3W and MaxVC by relieving the network manager from having to specify a info text filename for each switch individually. The legnth of the info text file must be less than 2K bytes after all the tokens have been replaced. Tokens are allowed in this file so that it can be automatically be customized for each switch. The tokens supported by NETMGR are as follows:

$A	Address	$C	Callsign	$D	Digi callsign	$F	Frequency
$I	DNIC	$L	Location	$N	Site name	$T	Time
$U	Date						

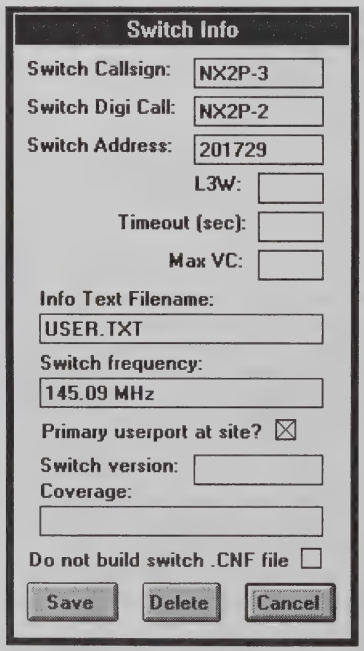
The info text file must be in the local NETMGR directory. This file can be created and edited using Notepad or the DOS edit command.

The **Do not build site .CNF files** check box is used to tell NETMGR that no .CNF files are to be built for this site. This allows a site to be treated as a "reference" site which is used to provide routing information for other sites during the build process without being built itself.

After all the above information has been entered, you can press the **Save** button which will save the information and close the dialog. You can access this information again any time by right clicking on the center of the site icon. The **Cancel** button lets you exit the dialog without saving any changes. Pressing the **Delete** button causes the site to be removed from the map and all links connected to the site to be deleted.

Defining Switch Information

Once the site information has been specified, you will want to enter the data for the switches at the site. Up to eight switches can be defined for each site/matrix. The switches are represented by the eight sections surrounding the center section of the site/matrix icon. To access a switch's information right click on the section representing the switch of intrest. In the case of entering the switch infromation for a switch that has not yet been specified, right click on one of the unused sections (best to select one on the side of the site/matrix icon that is the direction that links will be drawn). A switch dialog box something like the following will appear.



Switch Callsign is the callsign (including SSID) for the switch that will be represented by this section. Also used for the $C token in the info text.

Switch Digi Call is the digi callsign (including SSID) for the switch that will be represented by this section. Also used for the $D token in the info text.

Switch Address is the address for the switch that will be represented by this section. Not specifying a address or removing the address has the same effect as pressing delete. Also used for the $A token in the info text.

L3W overrides the default L3W that is entered for the site. If this field is left blank, NETMGR will use the site default.

Timeout overrides the default Timeout that is entered for the site. If this field is left blank, NETMGR will use the site default.

Max VC overrides the default Max VC that is entered for the site. If this field is left blank, NETMGR will use the site default.

Info Text Filename overrides the site default info text filename. See the site default section earlier in this document. If this field is left blank, the site default text will be used. In this case USER.TXT has been specified which is a general user port text file containing tokens that make it look like it was generated specifically for this switch.

Switch frequency is a text note for frequency. It is only used when building the info text by substuting whatever you enter in this field for any $F tokens used in the info text file.

Primary userport at site check box is used to tell NETMGR that this switch call and address should be displayed under the map and printed out with the map. This check box has no other effect.

Switch Version is currently unused for anything other than a place you can place this information to keep track of it. In the future, I plan on using the information to automatically call CONFIGUR.EXE of the correct version to go direct from NETMGR to .TBL files thereby avoiding a manual step. In order to matain compatability with the RZ series of programs (RZLOAD, RZCONFIG and RZSYSOP) I recommend you matain this data using the 6 digit version date stamp.

Coverage can be used to enter coverage addresses for a switch. A single line up to 72 characters can be entered which has any number coverage addresses seperated by spaces. NETMGR currently does not support "virtual" switches therefore the same coverage address must not be entered in more than one switch or else the address will appear more than once in each routing table. This WILL cause switches to lock up. If a virtual switch is being used, only put the coverage address in one of the switches and then manually edit the configuration files of the other switches by adding the coverage and deleting the covered address from the routing section of the configuration file.

Note - A virtual switch would be used in a case where two or more switches on the same frequency have some amount of common coverage. The common coverage area can be assigned a address and this address be put in the coverage of both switches. This allows any connects into this area covered by the virtual switch to route to the nearest switch thereby providing maximum effiency.

Do not build switch .CNF file check box is used to tell NETMGR that a .CNF file is not to be built for this switch. This allows a switch to be treated as a "reference" switch which is used to provide routing information for other switches during the build process without being built itself.

After all the above information has been entered, you can press the **Save** button which will save the information and close the dialog. You can access this information again any time by right clicking on the center of the switch section of the site icon. The **Cancel** button lets you exit the dialog without saving any changes. Pressing the **Delete** button causes the all the switch information to be removed. Links connected to the switch to be deleted.

Once the switch information has been specified you will note that whenever you move the mouse cursor over a section, the switches call and address appears in the ID windows. This provides a way of knowing what switches have been defined and a way of identifying the switches when you are adding the link information to the map.

Drawing Links

The next step in building the map, once you have at least two sites with at least one switch at each site defined, is to draw the RF links. To do this simply left click over a section on a site/matrix icon for which you have defined switch parameters. At this point a line will appear between that section and the mouse pointer. As you move the mouse pointer the line moves with it. Now move the mouse pointer over to the switch at the other end of the link. Left click on the section of the site representing the destination switch (note that links cannot be drawn within a site). To terminate the link draw process without creating a link, left click on the same site you started to draw the link from. As you draw links, you will note that some links will be green lines while others will become dotted red lines. When NETMGR sees that more than one RF link has been drawn from a particular switch, it changes from green to dotted red so that dedicated links (green lines) stand out as being virtually "collision free" (assuming there are no stations on the channel that are not shown on the map).

Each link has an associated Q factor, arbitrarily defined by the sysop. The lower the Q factor, the better the path. To set the Q factor for a link, right click on the line representing the link. This will cause a dialog to appear that lets you enter the Q factor. The Q factor is used in determining what the best and/or shortest path to a destination address and for alternative routing. The default Q factor is set to 1 whenever a link is added. Whenever there is a mix of backbone and 2m paths, it is a very good idea to set the 2m Q factors to a relatively high value such as 15. This means that for the routing to take a 2m path, there would have to be more than fifteen Q factor 1 backbone hops over the same path. Q factors can be any value from 0.01 to 1,000 per link. Fractional Q factors are allowed and are a good idea for high speed links. The way NETMGR builds the routing table is to look at all possible paths to a destination site and add the Q factors of all the links. The lowest total Q factor path is used as the primary path and the next to lowest total Q factor is used as the alternative route (note that second alternative routes have found to be generally ineffective. In practice this results in connects that seem to hang as the number of combinations get large. Thus, NETMGR only records a primary and first alternate path).

To delete a link, left click on the line representing the link. The line will change color to blue. Then under the edit menu select the delete link option. This will delete the link. If you decide you do not what to delete the link, just left click someplace on the background of the map. This will cause the line to return to its normal color.

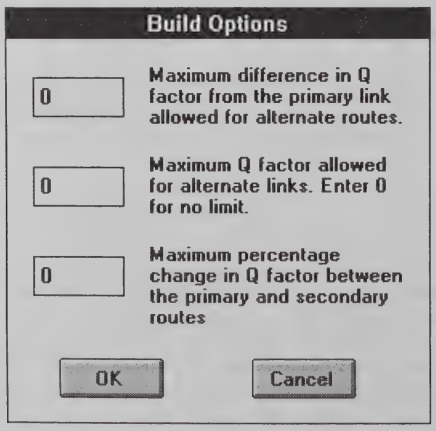
Building Configuration Files

Once you have entered all the site, switch and link information you are ready to build the configuration files. This is the easiest step in the process. Just select 'Compile' 'Build', which causes all the configuration files (.CNF) to be built. The status of the build process will be shown in a window. Any errors or warning will appear in the status window and, once the build process is complete, you can browse through them and make any fixes necessary. To close the status window, double click on the control box in the upper left corner.

The configuration files are saved in either the NETMGR directory or the directory from which you have loaded the map file from or previously saved the map file to.

Build Options

Several optional parameters are available to control generation of alternate routes. These are designed as a mechanism to prevent alternate routes from being created using paths which cannot support the traffic or alternates so lengthy they should never be tried. These parameters are saved with the MAP file. These options can be set by selecting 'Compile' 'Options' . The following dialog appears:



Maximum difference in Q factor from the primary link is used to prevent alternative routes from being generated which are more than this threshold higher in Q factor than this limit. This is useful in cases where there is a very long loop of switches. In this case NETMGR will see this and create the alternative routes (both ways around the

loop). At first, this appears wonderful, but the many hops required to get to the adjacent site 'the long way' will overload all the other switches. Using this control can prevent this while still allowing a alternative route to be created for the distant switches on the loop where the primary and secondary routes are nearly the same length. This control can be disabled by setting it to zero.

Maximum Q factor allowed for alternate links prevents alternative links from being generated when the Q factor of the alternative route exceeds this value. This is useful for preventing secondary routes from being created using paths which should never be used as a alternate (i.e. they can not support additional traffic). This can be done by setting this limit arbitrarily high (i.e. 9000), then you can set the Q factor for any links which are not be used to 10000. Caution should be used with this control. Setting it too low can result in no alternative routing being generated at all for long paths (i.e. paths where the primary link has a Q factor above this limit). This control can be disabled by setting it to zero.

Maximum percentage change in Q factor between primary and secondary routes prevents alternative links from being generated when the Q factor is greater than the specified percentage larger than the primary route. For example a primary route with a Q factor of 4 and this control set to 50%, the Q factor of the alternative can have a value up to 6. Extreme caution should be used with this parameter since this can prevent alternative routes from being created for very short paths. For example, when set to 40% and a primary route over a single hop path with a Q factor of 1 will require a alternative path with a Q factor 1.4 or less. If you have no paths with Q factors less than 1, no alternatives will be generated. This control can be disabled by setting it to zero.

Exiting

To exit NETMGR select 'File' 'Exit' or double-click on the control box in the upper left corner. Note that there is no checking of whether the map has been saved or not, so make sure you save your files before exiting.

Notes

1. NETMGR uses standard file NEW, OPEN, SAVE and SAVE AS ... commands for loading and saving the maps. (Note that there is no warning when you overwrite an existing file).
2. There is a trace option available for the build process. Selecting the trace option will result in a much longer build process. With this option selected you will see the links change color as NETMGR examines them. It makes a pretty good show.

Hints

1. To handle routing out of your management area, simply add a site to the map representing the first switch outside your area. Enter the proper callsign and address of this switch. Then in the coverage section enter all the address that need to be routed out through that link. If there is more than one link into the adjacent area, place one site for each of the links. You then enter the appropriate information in the coverage for the particular link. Do NOT duplicate information in the coverages as this will cause the problems by having duplicate listings in the routing table. If the links provide backup for each other place a link between them, this will cause the alternate routing to be built into the routing tables.
2. Avoid drawing very poor paths that might be used as alternate routes. In practice a poor quality alternate route usually will fail once loaded with the primary route's traffic. Therefore it really does not provide a backup path in practice. Each alternate path increases the amount of paths that must be examined by NETMGR, eliminating what would be useless links both aviods routing on useless paths and speeds up the NETMGR build process.

Conclusion

NETMGR is a powerful yet easy-to-use utility for configuring ROSE networks. The simple drag-and-drop construction lends itself to easy mapmaking, with the added benefit of clear and compact network documentation. The NETMGR software is shareware, and can be downloaded from CompuServe's HAMNET forum, or obtained by sending a SASE with a formatted 1.44 Mb floppy to the authors at the address above. As of this writing (July 1995), version 1.5 is current.

Graphical Information Systems and Ham Radio

(The future of A.P.R.S. technologies)

Keith Sproul, WU2Z
698 Magnolia Road
North Brunswick, NJ 08902-2647
(908) 821-4828 Home
ksproul@noc.rutgers.edu
http://www-ns.rutgers.edu/~ksproul

G.I.S. is a major buzz-word in the scientific and computer graphics communities. GIS, or Graphical Information Systems (also sometimes called Geographical Information Systems) is the display of significant amounts of data on a graphical system, usually a map of some kind,

GIS programs have historically been run on large systems, usually UNIX based systems due to the large amounts of computer horse power required. Over the last couple of years, this type of program has started to appear on desktop computers as these computers have grown in power and capabilities.

Typical applications of GIS programs have been to display data any where from the placement of telephone poles, to the habitat of animals, to highway and urban planning to pollution problems. This type of program typically sells for anywhere from $500 to $10,000.

Within the ham radio community, we have all seen the many different satellite tracking programs that are out there, and of course simple DX logging type applications that use maps. Although map-based, these programs don't really fall into the category of GIS applications.

Over the last several years, other GIS-like programs have started to be introduced into the amateur radio community.

• 1992 **APRS** [1] Automatic Packet Reporting System.

• 1993 **Mail Tracker** [2] a program that displayed where Packet BBS messages went as they traversed the country, displayed on a map overlay.

• 1993 **Packet Tracker** [3] Although not map based, Packet Tracker showed the logical connections between packet stations. This program was really more like a network analysis tool, but the graphics, the logical connections displayed and the large amounts of data collected made it have some of the features of GIS applications.

•1994 **MacAPRS** [4] the Macintosh version of APRS. This version of APRS has a lot more databases built into the program and uses maps with much more data points..

When **APRS** was introduced in 1992, it was a low-end GIS application. When the Mac version came out a year and a half later, **APRS** had evolved quite a bit from its original implementation, and now, a year later, **APRS** and **MacAPRS** have continued to evolve rapidly. **APRS** was originally developed to track the ships from the Naval Academy while on summer maneuvers. Since that time, it has evolved into a program to track weather, hurricanes, balloons, bicycles, and many other applications.

Maps

APRS depends on maps made either by hand, or more recently, maps made from USGS (United States Geological Survey) data. Early this year, a lot of the USGS data has become available for free on the Internet. This readily available data has created a surge in the interest in making maps for localized areas, and maps for special events.

A problem with the large amounts of map data now readily available is that the data is rapidly exceeding the capabilities of some of the lower-end computers found in ham-radio shacks. This is very evident in that the PC version of APRS which can only handle maps up to 3,000 points. The Mac version can literally handle maps of upwards of 1 million points.. although 10 to 30 thousand points is more typical of what is being used. The trade-off of this is that more points slows map redrawing down. With the newer desktop computers now available, this becomes less of a problem and people want better and better maps.

Databases

Other things that have been integrated into this type of programs is the ability to look up different pieces of information that is relevant to what you are working on. For example, the Macintosh version has the following databases built into the program:

ZIP CODE LOCATION Every zipcode in the country can be located.

AIR PORT LOCATION 18,000 US airports and 4,000 foreign airports are included. All of the US airports have altitude in addition to the lat/lon.

DXCC Database (See the discussion below).

Call Sign Databases MacAPRS supports the following call sign databases.

Buckmaster (all versions, this has the most extensive database)
Percon
QRZ
Amsoft

FCC Recently, the FCC has made a database that is updated weekly available on Internet. This database is a 30meg ZIP file that decompresses to a 130meg file. It is free to anyone that wants to take the time to download it.

All of these databases and others that are being worked on continue to take more and more memory and/or hard drive space. People are even copying the callsign databases to their hard drives for connivance and speed.

Specialized Features

APRS and **MacAPRS** lends itself quite readily to a wide variety of specialized applications. The authors of both programs have started to add many specialized features for different applications. Many people have used these programs for purposes that the authors never even thought of in the early days. Several of these are discussed below.

Weather Tracking One of the early uses of APRS has been to track weather. This use has been growing quite rapidly, especially in areas prone to experience bad weather. Both APRS and MacAPRS have added many features just for weather, and have added a lot of alarm capabilities to satisfy the Skywarn people. Both programs interface with weather recording equipment so that they can report the weather in real-time. The programs even support stand-alone weather stations for remote weather sensing.

DX Cluster For the person that wants to use a DX Cluster, APRS and MacAPRS will listen to the 'DX SPOTS' issued by a DX-Cluster System and plot the stations on a world map in the correct area. This is done by looking up the call-sign prefix in a database that says were they are located. The 'station' is then plotted on a map showing you where that station is. This brings an entirely different way of viewing this data, and you can tell graphically where the propagation is currently the best just by looking at a map.

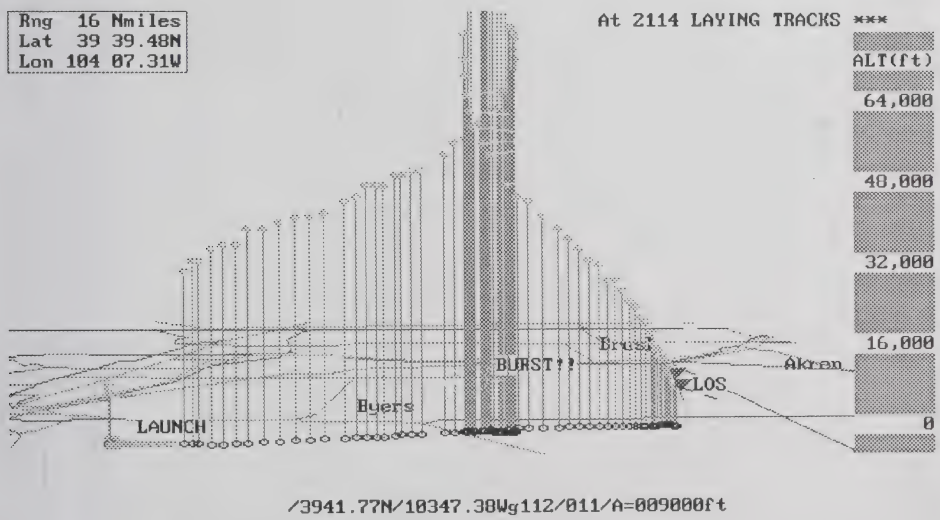
Direction Finding There is a lot of interest in Radio Direction Finding. There are two main groups of people that want this type of capability. The first group is people involved in search-and-rescue type operations, and the second group is people that want to track down repeater jammers or other types of illegal radio interference. The APRS programs work with automatic direction finding equipment and display the results on a map. They can also send the information to other APRS stations, and therefore allowing instantaneous triangulation showing where the radio signal is coming from. In near future, the Macintosh version will even be able to then give a list of all hams living within a specified radius of that point. This feature is just now starting to grow, especially since there is some low-cost Automatic RDF equipment starting to become available.

Meteor Scatter Recently, we have started to work on features to enable meteor-scatter communications via APRS. It would be very interesting to see where this type of communications actually will work, and APRS has proven to be a very good method for this type of research. It has many advantages in that it can be fully automatic, it uses very short packets, and it shows instantly where the communications are on a map. Some other people are also working on using APRS for E-Skip propagation testing.

Balloon Tracking. We have been doing work for balloon tracking. The PC version added a feature where it would show a pseudo 3-D display showing where the balloon was in relation to the ground. See Figure 1.

The Macintosh version has also added a several features to aid the balloon tracker to predict where the balloon will go BEFORE it gets there. This is based on some work done by Bill Brown, WB8ELK and several other people including some work by NASA and the military. The Macintosh version has added a separate graph to show altitude, ascent, and descent rates. See Figure 2. The numbers on the right hand side of the graph are the wind reports from the FAA. This data is used to predict where the balloon will land, based on how high it is expected to go, and how fast it rises. See Figure 3. The upper line is the PREDICTED path, and the lower line is the actual data. The predicted path shows the balloon landing only about 10 miles from where it actually landed. Unfortunately, the data from the FAA only goes to about 50 or 60 thousand feet, and therefore, the winds above that altitude are not known. In this particular balloon launch, some of the balloon chasers actually got to see the balloon come down!

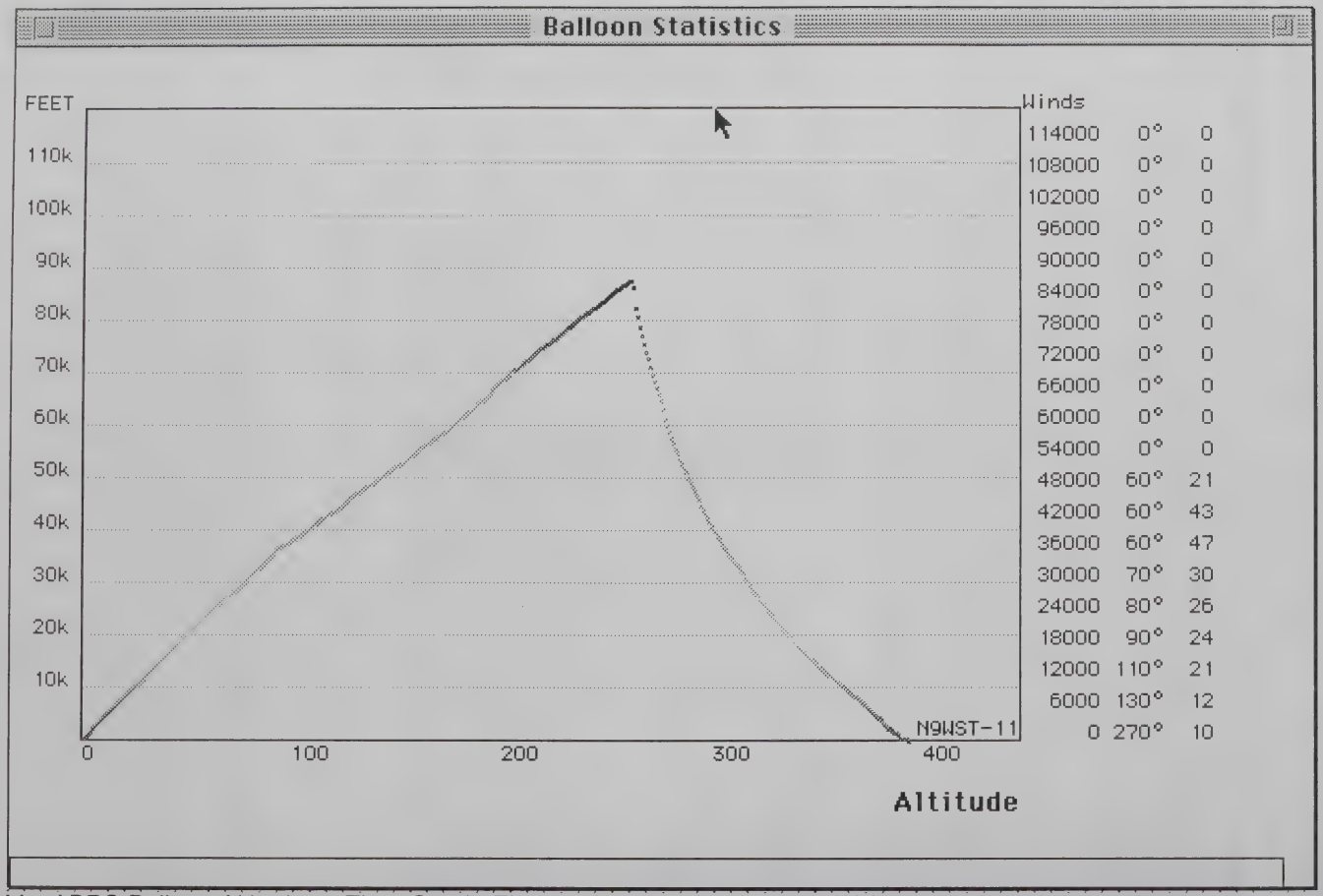
Figure 1



Pseudo 3-D display in PC APRS

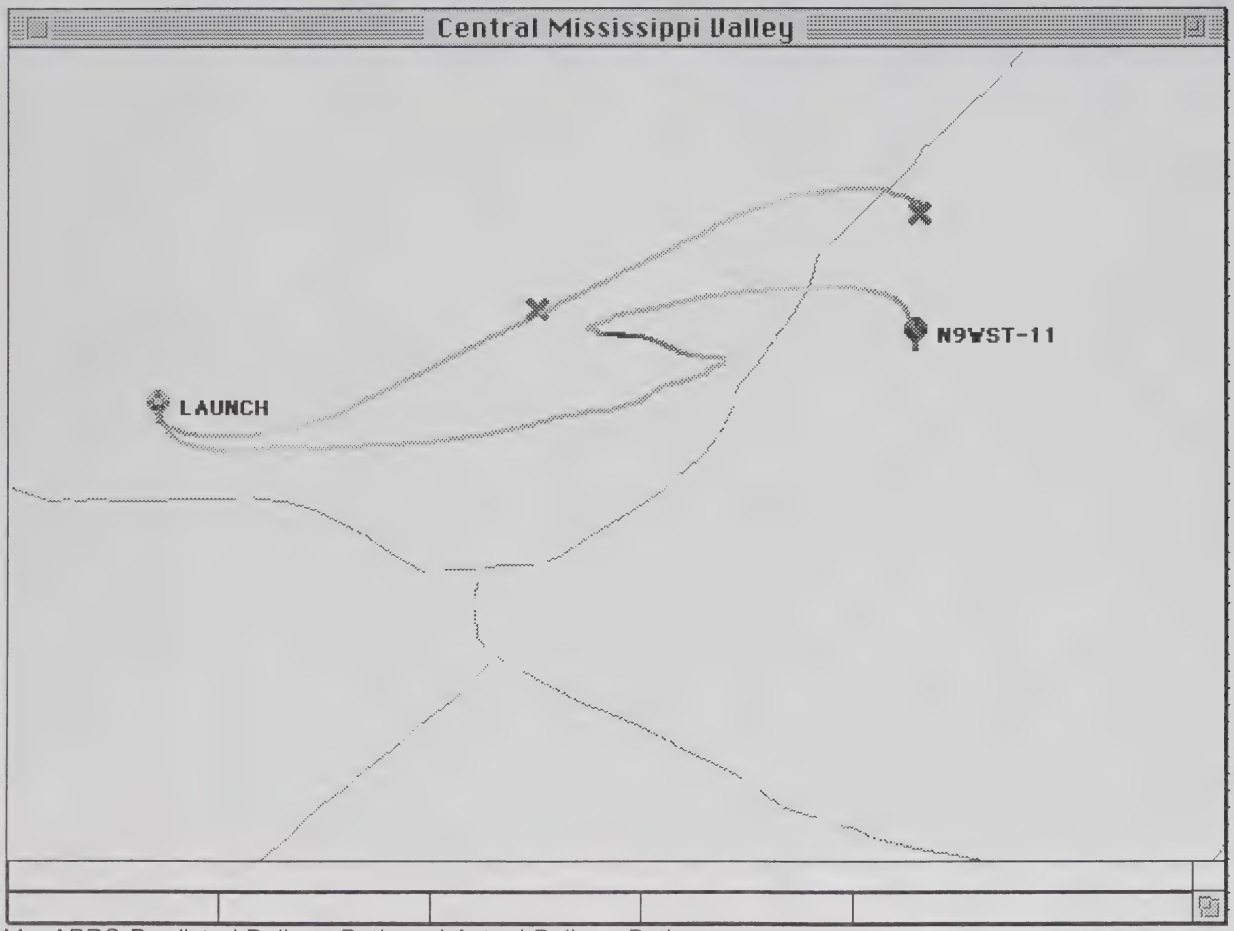
Model Rocketry Hams have been putting electronics up in model rockets for a long time. The most popular thing to do with model rockets is ATV. There are now groups of people doing high-powered model rockets going up 10,000 - 20,000 feet and even higher. With rockets going this high, it becomes desirable to track them too. Most of the software and features of the balloon tracking described above are used identically for rocket tracking, however, there are some minor differences used in the prediction of where the rocket will land. Also, due to the much shorter duration, the

Figure 2



MacAPRS Balloon Altitude vs Time Graph. The numbers on the right are FAA reported wind data values. (Actual altiude data for balloon in Figure 2)

Figure 3



MacAPRS Predicted Balloon Path and Actual Balloon Path
(Upper line is predicted path, lower line is actual.path)

data is generally transmitted much more often when tracking a rocket vs. tracking a balloon.

Three-D Graphics

With the introduction of the RISC PowerPC from IBM/Motorola/Apple, we now have a lot more power available to the desktop than was available even as little as a year ago. PowerPC computers are now being sold by Apple, IBM, Motorola, and several Clone manufactures. Along with this new CPU architecture, Apple has introduced a new set of graphics routines that run on the PowerPC Macintoshes called **QuickDraw-3D.** This set of routines allows programmers to 3-D graphics relatively easily.

As stated above, earlier this year, the USGS released a lot of their data on the Internet. Digital Elevation Data is one of the data sets available..

With the combination of these two items, we can now develop sophisticated 3-D graphics systems and can start to do many new things for Amateur Radio and map software.

The first application that comes to mind is a TRUE 3-D APRS system. See Figure 4. As you can see, with Quick-Draw 3D from Apple and the Digital Elevation Data from the USGS, we can do a quite sophisticated 3D system relatively easily. This type of system has its applications, especially if you live in a mountainous area. Although interesting and neat to look at, a 3-D system isn't always better than a 2-D system. The reason for this is that most people are used to looking at 2-D maps and adding the 3-D information would just lead to confusion.

We are developing a 3-D program that will do real-time 3-D visualization and allow 'tracking' through 3-D worlds from either live Packet data from the air, directly from a GPS unit while driving in your car, or from recorded data. (Check out the Web sites listed in at the end of this paper). An interesting application for this type of program will be to visualize what a balloon would see as it comes back to earth. There are some Quick-Time Movies up on the Internet showing this. These 'movies' can be downloaded and displayed on either Macintosh or Windows computers.

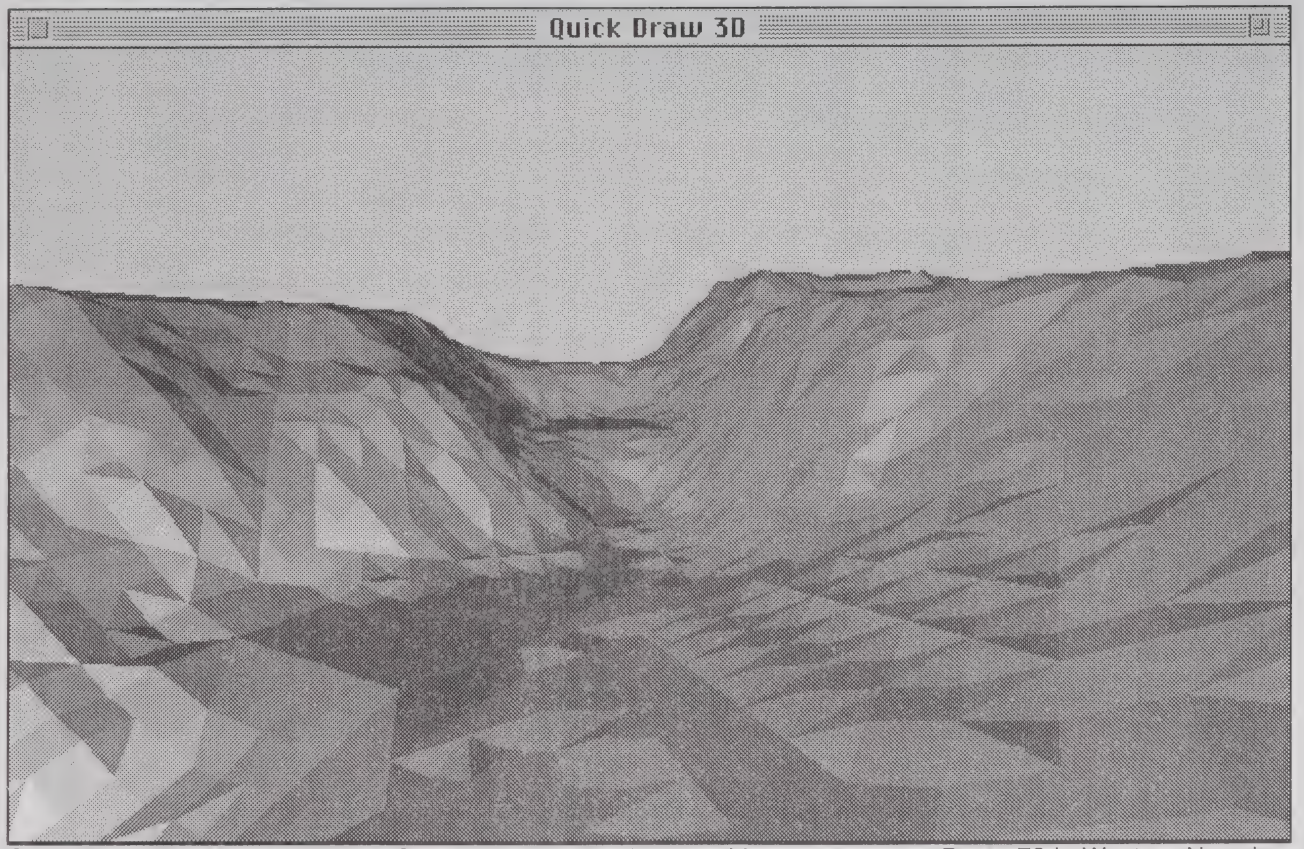
Another application that comes to mind is that you can now have a much better system for estimating repeater coverage. Yes, this type of software has been available for a long time, but in the past, it was very expensive, and also very slow, and generally only available to the 'commercial' radio people.

Large Databases

There are now large databases available, either public, or for low cost that contain HUGE amounts of data. A couple examples are

A named-geographic-location database from the USGS that contains 2 million named locations within the US.

Figure 4



Quick-Draw 3D software by Mark Sproul showing the Jugtown Mountain pass on Route 78 in Western New Jersey.

Percon's Spectrum database with all of the FCC licensed commercial broadcasting stations. Information like this is highly desirable for use in a ham radio related graphical information system.

These types of databases will become more and more readily available to the ham community and we will want to take advantage of the data available to us. We need software that can handle them efficiently and customized for our needs.

Future of GIS software

A common comment within the education and programming communities is:

"Don't let lack of computer power stop you from writing something because by the time you get it written, there will be computers with enough power to do it".

With the introduction of RISC technology, and the PowerPC, and to some degree the Pentium, people are getting lots more powerful computers at home and in their ham shacks.

The following was taken from the Internet newsgroup **comp.infosystems.gis**

Subject: Re: GIS around the year 2000
Newsgroups: comp.infosystems.gis

Question
So, does anyone want to get out on a limb and suggest what sort of hardware and software used in GIS will be common in the year 2000?

Answer
Well, I've been around long enough to remember the same history. I think people tend to overestimate the speed things will happen, but here's my best shot at what I think I'll be doing 5 years from now.
On broad themes, integrated raster/vector databases will be common. Databases will range in the 100s of GB in size for the mid-to-high end users. Terabyte databases will be common at the highest end, with hyper-spectral remote sensing data being the biggest resource hog. Hardware will have a hard time keeping up with the data volumes. Real-time differential GPS will be ubiquitous, and the spatial accuracy/resolution requirements for GIS databases will tend toward the sub-meter and centimeter level, because the technology will allow it. (Most current applications I deal with don't need that kind of spatial detail, but as soon as it's available, people will decide it's necessary.) The fields of surveying and GIS will be difficult to differentiate, too, as a result of the increasing spatial accuracy requirements. Professionals in both fields have a lot of learning to do in the next five years, to be able the master these currently very different fields. Projections, datums, geoids, and network adjustments will become considerations for every project. People will start worrying about the affects of tectonic movement on the quality of their databases.
--
Alan Rea, Hydrologist
ahrea@csdokokl.cr.usgs.gov

U.S. Geological Survey:
202 NW 66th ST, Bldg 7
Oklahoma City, Oklahoma, USA 73116
(Note: This is his personal opinion and not that of the USGS)

What gets interesting and exciting about this type of capabilities is where will it lead and what will be available to the ham radio operator as the power of the our computers increases and as the software gets better.

Conclusions

GIS technologies are changing rapidly. They are pooping up in more and more applications in our day-to-day life. Faster and bigger desk-top computers are allowing this type of applications to migrate to the individual user. This type of software has many applications within the ham radio community. We need to be aware of this type of software system and take advantage of it. We also need to realize that we can't continue to support the lowest common computer around for ever, i.e. 8088 or even 80286, and 80386.

The capabilities available to us are growing faster than we can keep up with. The only real limitation is our imagination and occasionally our pocket-book, but the possibilities are limitless, and the future, even the very near future looks very exciting!

References

[1] ***Automatic AX.25 Position and Status Reporting***, Bob Bruninga, WB4APR, ARRL 11th, Computer Networking Conferences, Teaneck, New Jersey, November 1992, pp 13-18.

[2] ***Mail Tracker, A Graphical Mail Tracking Program***, Keith Sproul, WU2Z, ARRL 12th Digital Communications Conference, Tampa, Florida, September 1993, pp 83-96.

[3] ***Packet Tracker, A Graphical Packet Tracking Program***, Mark Sproul, KB2ICI, ARRL 12th Digital Communications Conference, Tampa, Florida, September 1993, pp 77-82.

[4] ***MacAPRS: Mac Automatic Packet Reporting System, a Macintosh Version of APRS***, Keith Sproul, WU2Z and Mark Sproul, KB2ICI, ARRL 13th Digital Communications Conference, Bloomington, Minnesota, August 1994, pp 133-145.

Internet Sources of Information

APRS and MacAPRS can be downloaded from:

ftp.tapr.org in the directory /tapr/SIG/aprs/uploads/

WWW Sites of interestes

APRS info

http://www-ns.rutgers.edu/~ksproul/MacAPRS

http://www.ccnet.com/~rwilkins/aprs.html

http://www.mindspring.com/~rwf/aprs.html

Apple Quick-Draw-3D Info

http://www.info.apple.com/qd3d/QD3D.HTML

http://www.njin.net/~msproul/macintosh/QD3D.hml

USGS Data

http://info.er.usgs.gov/data/index.html

AX.25 Transport Layer Drivers for TCP/IP

by
Mark Sproul, KB2ICI
Tim Hayes, N2KBG

sproul@eos.ap.org
thayes@noc.rutgers.edu

Abstract

Current TCP/IP over packet radio is largely implemented with KA9Q's NOS and its many variations on PCs and NetMac on Macintoshes. NOS was written before good general purpose networking software was available and was written as one monolithic do-everything program. Today's multitasking operating systems have built in networking support and there exists a large base of good server and client software for all platforms (Macs and Windows) that use this networking ability. Because of these things, a large, monolithic program is no longer needed and is actually a severe handicap. By implementing a transport layer AX.25 driver for use with a system's native TCP/IP protocol stack, the existing programs that utilize TCP/IP for Internet activities can be used for amateur packet radio activities. These include many excellent free and shareware programs for both Macs and Windows machines.

Background

NOS was written in 1985 [1] to run under DOS which does not allow multitasking at all. NOS was written as its own multi-threaded environment to implement all of the various functions needed in a network station including mail server, file server, router, user interface for these functions etc. Because DOS is not multitasking, NOS normally requires a dedicated machine to run and NOS is very complicated to set up. By today's graphical user interface standards and multitasking operating system environments, NOS is obsolete. In addition to being obsolete, there are so many different versions of NOS, each one to fix or address different issues, it is impossible to keep track of.

On the Macintosh platform, NetMac was a port of NOS to the Mac operating system [2]. NetMac suffers from many of the same problems as NOS and it has never been up to normal Macintosh standards for user friendliness.

Software Requirements for a typical TCP/IP packet radio station

A user participating in an amateur radio TCP/IP network needs software to fulfill a basic set of functions to successfully participate. In the past, most of these functions have been accomplished with NOS. With the advances in the current TCP/IP networking software, the capabilities of NOS have fallen way behind those of most systems with access to the Internet and it will never catch up in its current implementation. If we could utilize existing software packages (see tables below) we would be able to immediately overcome these limitations. In addition, we would not have to reinvent the wheel each time a new network feature became available; software packages created for use on the Internet could be immediately put into service by amateurs over packet radio.

Server software

Function	Mac	Windows
Domain Name server	MIND	
IRC	Chat	
FTP server	FTPd, FTPShare	EMWAC FTPd
Gopher server	FTPd	
ListServer	Macjordomo, AutoShare ListSTAR	MajorDomo (NT)
Mail server	MailShare(AIMS), MailStop, MacPost	EMWAC POP3 EMWAC SMTPd
News Server	???	
WWW server	WebStar, MacHTTP, FTPd	EMWAC, NCSA-HTTP NetSite (NT)

Client software

Function	Mac	Windows
Archie		WSArchie, Prospero
FTP	Fetch, Anarchie, XferIt	Chameleon, FTP
Gopher	TurboGopher	WSGopher
IRC	ircle	WinTalk, WSIRC
Mail	Eudora, PopMail, BlitzMail lee-mail, MacPost,	Eudora,

	Mail-Connect	
News	NewsWatcher, Nuntuis	WinVN,
	NewsHopper, NewsSucker	
	InterNews	
	Newsview, MacSlurp	
Terminal	NCSA Telnet	NCSA Telnet, EWAN
WAIS	MacWAIS	
WWW	MacWeb, Mosaic, Netscape	Mosaic, Netscape,

Implementation

How do we utilize all of this existing software without re-writing it? We have to write a transport layer for TCP/IP that implements AX.25. Once this is implemented, all of the current TCP/IP client software and server software for Macs and Windows is immediately usable on the packet radio TCP/IP - AX.25 network.

Status

Currently under development by the authors of this paper is an implementation for the Macintosh. It is called MacAX.25 and is a transport layer driver for use with MacTCP. Once finished it will be able to accomplish all of the goals outlined above. The current projection is that it will be finished by the end of 1995 or sooner.

There is also a project to implement an AX.25 TCP/IP driver for Windows, is in progress called EtherAX.

Linux already has an AX.25 driver for use with its TCP/IP stack.

MacAX.25 Design Specifications

SCOPE:

This document describes an AX.25 driver for IP stacks. All AX.25 procedures conform to the ARRL's AX.25 V2.0 specifications

DESIGN SPECS:

- Uses Generic C Code

For easy porting across platforms. In its final form this driver should be cross-platform compatible. Except for routines to pass data back and forth between the IP stack and this drive no code should be OS-specific.

- AX.25 Transport Method
  AX.25V2 UI frames will be used to send datagrams

- IP encapsulation
  IP datagrams will be directly inserted into the data field of the AX.25 packet.

  The PID of the AX.25 packet will be set to "0xcc"

- ARP encapsulation
  ARP datagrams will be directly inserted into the data field of the AX.25 packet.

  The PID of the AX.25 packet will be set to "0xcd"

- ARP handling on the client system
  This driver should use the native IP stack of the client system for handling ARP information. This means it is necessary to encode AX.25 addresses in a manner that the system can store in its ARP tables.

  This encoding should be direct if possible- the AX.25 address field to be stored should have it's C and R bits cleared (if set) and all 56 bits stored in the ARP table.

  In the event that the IP stack's ARP table can not be used to store AX.25 addresses directly they should be encoded and stored as an ethernet MAC address. In this case the 56-bit AX.25 addresses should be encoded into the 48 bit MAC address as follows:

```
rrrrrr p cccccc cccccc cccccc cccccc cccccc cccccc ssss
MSB                                                 LSB
<----------------------- 48 bits ---------------------->
```

  - The most-significant 7 bits are reserved and should be set to ones.
  - The "p" bit should be set if an AX.25L2 path is in use for the address. In all other cases it should be cleared.

- The next 6 sets of 6 "c" bits each (36 bits total) contain the letters of the uppercased callsign translated to the offset of the character from an ASCII space. Empty positions should contain an translated ASCII space (ie: 0x00).
- The least significant 4 bits contain the SSID of the address which are not translated in an way.

Any published ARP entries must be published in the encapsulated form.

It is not permissible to publish ARP entries for addresses which have AX.25L2 paths.

• IP should handle datagram delivery and routing, not this driver
Simple and complex mechanisms for doing dynamic AX.25L2 routing of AX.25 encapsulated IP datagrams have been developed. None of these should be implemented or used by this driver. All routing should be handled by IP.

A mechanism should be implemented to allow user-defined point-to-point paths at the AX.25 level for links beyond the reach of the local transmitter. This may be especially useful when the "local gateway" is a hidden transmitter.

- This is not routing- a path defines a single static link much as a telephone number defines a dial-up PPP link.

- The AX.25 driver will be responsible for managing the paths.

- Any AX.25 path mechanism should be capable of being used by both ARP and IP.

• Plain AX.25 L2 Support
There will be no support for vanilla AX.25 connections

As connections are not implemented the driver will not be able to accept AX.25 SABM requests. SABM requests from other stations should be handled in accordance with the AX.25L2V2 spec. (All incoming SABM requests should be immediately replied to with a DM frame in which the Poll/Final bit is set)

Incoming I frames will be dropped with no response.

Incoming S frames other than those of type SABM will be dropped.

Incoming UI frames with PID's other than those for IP and ARP will be dropped.

URLs:

Authors Home pages

http://pilot.njin.net/~msproul

http://www-ns.rutgers.edu/~thayes

AX25 specs

http://www-ns.rutgers.edu/~thayes/ax25

http://www-ns.rutgers.edu/~thayes/ax25/MacAX25

http://www-ns.rutgers.edu/~thayes/ax25/MaxAX25/MacAX25spec.html

Mac Software

http://rever.nmsu.edu/~elharo/faq/software.html

http://leuca.med.cornell.edu/Macjordomo

[1] ***The KA9Q Internet (TCP/IP) Package: A Progress Report***, Phil Karn, KA9Q, ARRL 6th Computer Networking Conference, pp 90-94, Redondo Beach, California.

[2] ***KA9Q Internet Protocol Package on the Apple Macintosh***, Dewane Hendricks, WA8DZP, Doug Thom, N6OYU, ARRL 8th Computer Network COnference, Colorado Springs, Colorado, pp 83-91

The Puget Sound Amateur Radio TCP/IP Network

Steve Stroh N8GNJ

Puget Sound Amateur Radio TCP/IP Group
14919 NE 163rd Street
Woodinville, WA 98072 USA
e-mail: strohs@halcyon.com

Abstract

The Puget Sound Amateur Radio TCP/IP Network (also known as WETNET, the Washington Experimenter's Tcp/ip NETwork), centered in the Seattle, Washington metropolitan area, has built an extremely functional packet radio network based on TCP/IP networking and cellular RF techniques. The network encompasses more than eighteen separate Local Area Networks, an estimated 200 users, four 9600 baud bit regenerative repeaters, and a full time Internet gateway. This paper is intended to provide an overview of an operational Amateur Radio TCP/IP network.

A Frequently Asked Question is "Why TCP/IP? Can't you do everything with regular packet radio that you can do with TCP/IP?" The answer is... somewhat, but not really. A key feature about Amateur Radio implementations of TCP/IP is that its various capabilities are built in- you don't have to combine dissimilar systems to do mail forwarding, file transfers, provide Packet Bulletin Board System (PBBS) services, multi-connect chat sessions, multiple ports, etc.- "It's (all) in there". A typical TCP/IP station can do:

- file transfers (including binary files)
- keyboard to keyboard (chat, telnet)
- finger (display short text files)
- electronic mail
- automatic routing
- ping (test link integrity)
- operate as a very capable Packet Bulletin Board System
- operate as a very capable Net/ROM, TheNET, X1J node, etc.
- accept multiple connections from AX.25, Net/ROM, and other TCP/IP stations
- access multiple ports, including modem, RS-232, terminals, and Ethernet connections

These operations are simultaneous, since the TCP/IP software is written to multitask. The capabilities outlined above are only a subset of TCP/IP's capabilities.

Some Background on Amateur Radio TCP/IP

A complete discussion of all of the various TCP/IP utilities and capabilities is beyond the scope of this paper, but a few deserve some discussion. One of the most interesting capabilities of Amateur Radio TCP/IP is e-mail. Since each TCP/IP station can send and receive e-mail, there cannot be a "choke point" in the network for message traffic- each TCP/IP user has the ability to send e-mail to any other user from their station. This was a much appreciated feature after having been through "PBBS Forwarding Battles", where certain PBBS Sysops wouldn't forward to other PBBS Sysops. With the added capability of a mailing list- being able to forward a single message to multiple recipients, e-mail made PBBS', and their problems, almost irrelevant. TCP/IP users have the capability to participate in PBBS forwarding, and can "translate" PBBS messages to e-mail messages, and vice-versa.

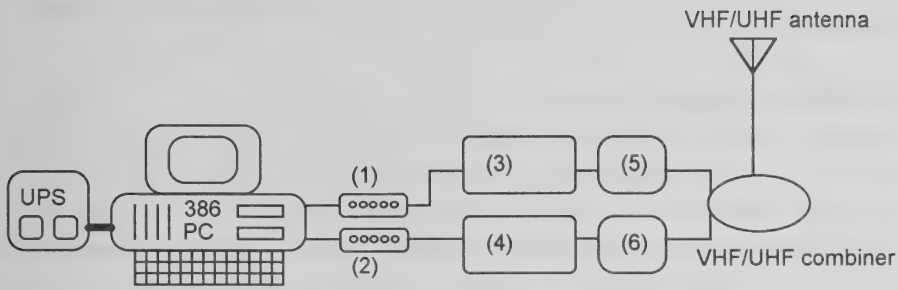
Being able to add multiple ports easily is liberating. TCP/IP users can gateway between multiple ports, and this capability is used to provide redundant routes, and to provide network access to remote Local Area Networks whose users cannot access other parts of the network directly.

A TCP/IP station can effectively service AX.25 users by providing the capabilities of a PBBS, a network node, a "chat node" , and much more to AX.25 users.

Amateur Radio TCP/IP authors have traditionally (begun by Phil Karn KA9Q) made all source code, (written in C) freely available. If you want to add a new feature, or fix a bug, the source code is available, and several Puget Sound TCP/IP operators have made good use of this to add features and fix bugs, and contribute back to the code pool.

Network Operating System (NOS) was the name that KA9Q gave to his "second generation" of Amateur Radio TCP/IP software (the first was named NET). Since NOS was released, there have been numerous offshoots of NOS, including JNOS, TNOS, and others. In this paper, NOS is used interchangeably with the term "Amateur Radio TCP/IP Software".

It is important to remember that "Amateur Radio" TCP/IP is completely interoperable with "commercial", or "wired" TCP/IP. TCP/IP was enhanced to make use of Amateur Radio just as it was to make use of satellite links, other networking systems, and modems.



PC runs modified NOS or Linux
(1) TNC w/TAPR 9600 modem w/bit regen (5) UHF duplexer
(2) TNC w/1200 baud w/open squelch mod (6) VHF RF eqpt.
(3) UHF land mobile radio, modified for repeater and 9600 baud
(4) VHF land mobile, modified for 9600 baud (used for 1200)

Figure 1 - Diagram of a typical Puget Sound Network Switch site

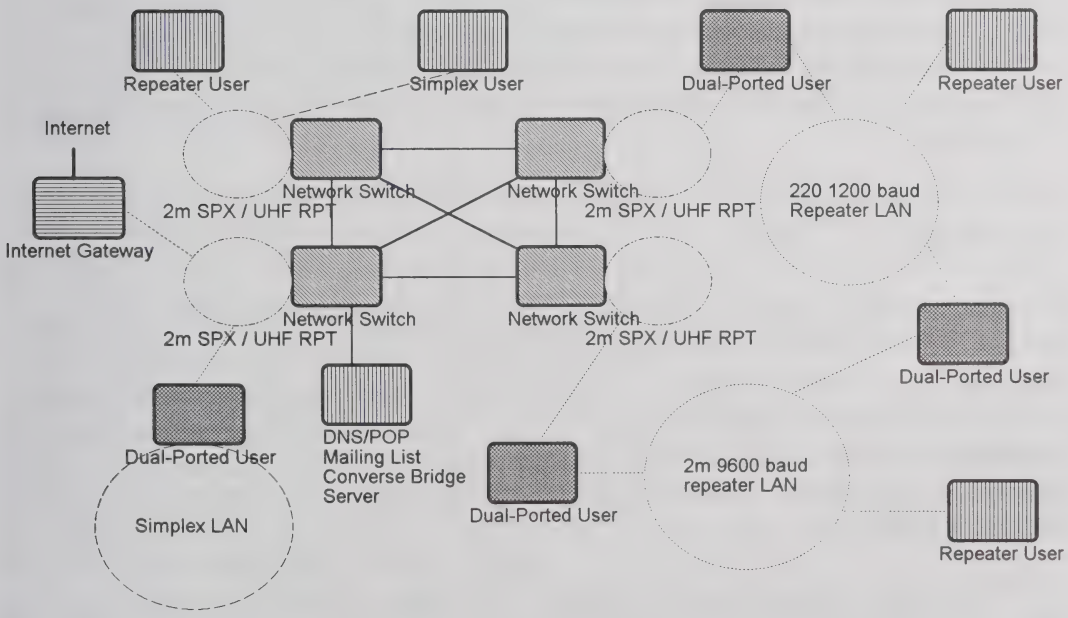


Figure 2 - Puget Sound Amateur Radio Network

Diagram of Potential Routes

Typically, the Switches do not offer any "user functions" and only handle routing and other network functions. If a user function is enabled, it is typically routed to other systems in the network. This may change as the Switches are converted to Linux.

Several LANs have been formed by multi-ported stations that access a repeater and a 2m simplex frequency. Network users have gradually adopted the cellular telephone approach of using low power and a low RF profile, and thus are able to reuse frequencies. Typically, the multi-ported station forms the center of a simplex LAN. Any stations joining the LAN are encouraged to be within good DCD range of the other stations on the LAN. This keeps overall performance on the LAN reasonably high, despite the inevitable collisions and hidden transmitters resulting from simplex operation.

The network is linked full time to the Internet through a Network Switch at the premises of one of the largest Internet Service Providers in the Seattle area. The primary use of the gateway is to allow participation in the global Converse Bridges. Amateur Radio TCP/IP users can also telnet, FTP, etc. to Internet hosts, as long as the user has a good connection to the network so the Internet system doesn't time-out from long delays.

The group makes extensive use of subnetting- the third "octet" of the Amateur Radio TCP/IP IP address is specific to a particular LAN. For example, 44.24.103.xxx is an IP address for the 220 1200 baud repeater, and 44.24.101.xxx is an IP address for the 147.60 simplex LAN. RIP (Routing Information Protocol) is used to propagate routes throughout the network.

Operational

A typical user station consists of a 386 PC using a Tekk KS-900 UHF radio with a beam, an AEA PK96 or MFJ1270 TNC with TAPR 9600 baud modem. The favored 2m 9600 baud radio is a modified GE MVP. The TCP/IP software of choice is the JNOS (WG7J). It is not uncommon to for the NOS PC to be configured as a "router" for other PC's in the household via serial or Ethernet connections to a second PC running "commercial" TCP/IP software, often with a graphical user interface- Windows/Winsock, OS/2 Warp, or Xwindows on Linux.

AX.25 users are welcome to connect to any TCP/IP station, both to browse around on that particular system or be routed to another system on the network. Some user stations and Switches actively encourage AX.25 and telnet connections by having numerous information files, binary files, and BBS areas available, . The group, at present, hasn't placed a high priority on interconnection between AX.25 PBBS forwarding and Net/ROM -type networking. There is limited connectivity between local PBBS systems and TCP/IP

users through a TCP/IP to PBBS gateway which transfers PBBS messages via e-mail. Personal PBBS messages are sent privately, and PBBS bulletins are sent via a mailing list.

Setup of TCP/IP for the first time has traditionally been a problem. The group has developed a JNOS setup disk, which includes all executable and configuration files necessary to get JNOS up and running. It includes a customized setup program which prompts the user for IP address, callsign, and other particulars, and then customizes the AUTOEXEC.NOS startup file, creates directories, and installs files as appropriate. From all accounts, the disk is working well in getting TCP/IP newcomers on the air.

The Puget Sound Amateur Radio TCP/IP Group

Early members of the group had been active in other area packet radio groups and had become burned out from the formal duties of a group, such as holding office, constitutions, dues, newsletters, etc. A deliberate, conscious decision was made that this group would remain informal in order to concentrate on technical and other innovative lack-of-management techniques. When funding has been needed for repairs or new equipment, money is donated by members of the group. This arrangement has worked relatively well, but the group did recently decide to get "slightly less disorganized" by requesting volunteers for *Keeper of the Notes*, *Keeper of the List of Projects*, and *Keeper of the Shekels*.

The group communicates primarily through a mailing list. Members of the group that wish to pose a question, comment on a previous posting, or share information send a single message to the list, and that message is automatically propagated. If a user's connectivity is good, messages from the list can be delivered in minutes. Mailing list functions have been automated using Majordomo mailing list server software on a Linux system.

The group meets monthly on a weeknight evening. In keeping with its informal nature, meetings are "moderated" only to the point of "collision avoidance". There is usually plenty to discuss, and formal presentations are rare. Any "burning issues" have typically been discussed on the mailing list prior to the meeting, and there are only a few "reports". The group has a small advertisement in the local "Computer Paper" which draws in a few lapsed hams and technically curious non-hams.

An ongoing effort is to write effective documentation. Those involved in the documentation effort are enthusiastic about the potential of using the World Wide Web (WWW, or Web) to distribute information. Articles can be distributed as they are created, and updated individually. Web pages, being composed of ASCII text, can easily be distributed as e-mail, or printed from a Web browser. Using Web pages also enables other

groups worldwide to easily access the group's information. The group will support Web pages on the Internet gateway (typical graphically rich Web pages), and a well connected RF site in the network (primarily text Web pages). The group's current Internet Web page is located at:
`http://wetnet.wa.com/`

Future goals and projects

- Enhancing the reliability of the overall network is a high priority
- Expand high speed TCP/IP links to other areas in the Pacific Northwest, especially Vancouver and Victoria, British Columbia and Portland, Oregon.
- Implement several 56 KBPS and higher links
- Replace NOS with Linux at all Network Switches
- Increased use of Linux and other TCP/IP capable systems
- Contribute to the forthcoming TAPR TCP/IP book
- Sponsor a TAPR Annual Meeting and/or an ARRL Digital Communication Conference
- Accommodate increased use of PC's, Macs, Amigas, and Unix systems running their native implementations of TCP/IP
- Aggressively test and implement new radio-based routing techniques as they are developed

Conclusion

TCP/IP works very well for Amateur Radio networking. It has been a satisfying learning experience to get it running and keep it functioning. The results are well worth the effort expended.

The group would enjoy hearing from Amateur Radio TCP/IP users around the country and the globe that have implemented Internet access to their Amateur Radio Networks.

For those that are able, wouldn't it be more interesting to use our "radio" computers to participate in mailing list discussions such as NOSBBS, NETSIG, and others? There is no reason not to if your system has adequate connectivity to the Internet.

References

Steve Stroh N8GNJ, 206-481-5735, strohs@halcyon.com for questions on the Puget Sound Amateur Radio TCP/IP Group,

Ken Koster N7IPB, kenk@wetnet.wa.com for questions on the Puget Sound Amateur Radio TCP/IP Network,

Terry Conboy N6RY, 206-450-8388, tconboy@uswnvg.com for questions on the US West NewVector Amateur Radio Group and the Cellular Base Station systems.

The most complete book on Amateur Radio TCP/IP is NOSintro, subtitled TCP/IP of Packet Radio, An introduction to the KA9Q Network Operating System by Ian Wade G3NRW, copyright 1992 by Dowermain Ltd., ISBN 1-897649-00-2. NOSintro is available from the ARRL.

A good FTP site for downloading Amateur Radio TCP/IP software from the Internet is:
`ftp.ucsd.edu/hamradio/packet/tcpip`

The mailing list for Puget Sound Amateur Radio TCP/IP Network users is:
`seatcp@algedi.ampr.org`

My thanks to the users and builders of the Puget Sound Amateur Radio TCP/IP Network- it's not a network without users to use it. It's been quite a ride, and we're "finally having fun yet". I'd especially like to thank N7IPB, KD7NM, WA7QFR, N7NKJ, and WA7FUS for inviting me into the group.

6PACK – a "real time" PC to TNC protocol

by Matthias Welwarsky, DG2FEF @ DB0KLN.DEU.EU, Am Pelz 77c, 64295 Darmstadt, Germany
translated by Tom Sailer, HB9JNX @ HB9W.CHE.EU, Weinbergstraße 76, 8408 Winterthur, Switzerland

During the development of the PC/FlexNet software package, there was a strong desire to use the existing hardware, especially the very widespread TNC2, which populates almost every packet radio station nowadays. Sysops of TheNetNode digipeaters also showed much interest, since many TNN nodes use TNC2 devices connected using a KISS token ring.

Existing TNC protocols have severe disadvantages. The KISS protocol does not allow much influence on the channel access. This prevents the implementation of alternative channel access schemes, such as DAMA or OPTIMA. The so-called "WA8DED Hostmode" does not provide data transparency, and its throughput is very limited.

The basic concept of 6PACK was developed at the end of 1993 by Ekki Plicht, DF4OR, Henning Rech, DF9IC and Gunter Jost, DK7WJ. It was further developed for use in the PC/FlexNet software package by Gunter Jost, DK7WJ, and Matthias Welwarsky, DG2FEF. The protocol then did not support multiple TNCs connected in a ring topology. It only allowed for one TNC per asynchronous serial interface. It was then found that a ring topology required different protocol features than a simple point to point connection.

The current revision of the protocol has many additional features compared to the first version. It now allows a ring topology of up to eight TNCs connected to a single asynchronous serial interface of the computer. The ring wiring is hardware compatible with the existing token rings used by the "TheNetNode" software.

6PACK provides:

- Data transparency
- predictable capacity requirements on the ring
- data and realtime information is distinguished
- fast response to changing channel usage, even under high loading and multiple TNCs on the ring
- automatic ring setup replaces a channel number "patched" into the (EP)ROM
- data is protected by a checksum

The asynchronous serial interface parameters are: 8 data bits, 1 stop bit, no parity bit.

The baud rate on the ring must be at least twice the highest HDLC bit rate of the TNCs connected to the ring. RTS/CTS is ignored, RTS however should toggle about every 10ms to reset an eventually connected hardware watchdog. The TNC software should implement another watchdog which terminates any transmission if no further data is received from the PC. To reset this watchdog, any command sent to the TNC will do, for example an LED control command.

Bits 6 and 7 of every byte distinguish between channel data and control codes.

76 543210	meaning
00 xxxxxx	channel data
01 xxxccc	status/protocol data
1x xxxccc	status/protocol data
x	data
c	channel address

Control codes always consist of only one byte. Code groups of more than one byte are never used. Therefore, control codes require little transmission capacity. Control codes are completely independent from previous or following control codes. This makes the protocol very robust. An exception to this rule is, as in the KISS protocol, the "start/end" command, which always belongs to a data packet, and the channel data itself. Because two bits are used to distinguish between channel data and control codes, only six bits per byte are available for channel data. Therefore, three data bytes have to be encapsulated into four "6PACKs", according to the following scheme. Note that a channel data byte does not correspond directly to a byte on the asynchronous serial interface.

	1st 6PACK	2nd 6PACK	3rd 6PACK	4th 6PACK
Codes	00xx.xxxx	00xx.yyyy	00yy.yyzz	00zz.zzzz
Bits	54.3210	76.3210	76.5410	76.5432

The symbols "x", "y" and "z" with the corresponding bit numbers represent the channel data. This rather complex scheme was chosen since it requires the least possible shift operations to extract the channel data from the 6PACKs.

The fast transmission of real time events is achieved by using priority codes. The TNC for example transmits an appropriate command code at every change of the DCD line. Additionally, the TNC sends the DCD state about every second to allow recovering of a potentially lost command code.

The coding of the command codes is shown in Table 1. The code 0xC0 (1100 0000) is not used in 6PACK; this should prevent an accidentally connected KISS TNC from sending.

The data transmission between TNC and PC

The following simple rules govern the data transmission between the TNCs and the PC.

- Channel data is always transmitted in whole packets. Every packet is enclosed with a pair of "start/end" commands. The channel number carried in the starting "start/end" command defines the source or destination of the packet. The first data byte carries the TX delay, either the TX delay that should be used by the transmitter, or the one measured by the receiver, depending on the data direction. The last data byte carries the checksum. It is adjustet so that the sum over the whole packet (including TX delay and checksum byte) plus the channel number add up to 0xFF (8bit two's complement arithmetic)
- Priority commands must be processed before ordinary data. If a TNC receives a priority code with an address that does not match its own, it must immediately retransmit the command, even if it is just sending a packet.
- A packet of channel data must not be interrupted by another packet of channel data. The TNC may only place its own packets onto the ring either before or after packets from other TNCs.

Each packet of channel data has the following structure:

SOF	TXD	DATA	CS	SOF

SOF Start/end of Frame
TXD TX delay in units of 10ms
DATA Channel data
CS Checksum byte

This structure results after extracting the data from the 6PACKs. The TX delay can be reconstructed only after the second 6PACK with channel data.

H—— ——L	command	direction
0100 0ccc	start/end of frame	PC ↔ TNC
0100 1ccc	TX underrun	TNC → PC
0101 0ccc	RX overrun	TNC → PC
0101 1ccc	RX buffer overflow	TNC → PC
011x yccc	LED: STA=x; CON=y	PC → TNC
10xy zccc	priority message	
	x = TX counter + 1	PC ↔ TNC
	y = RX counter + 1	TNC → PC
	z = DCD state	TNC → PC
1100 0000	not used (KISS FEND)	
1110 0ccc	send calibration pattern for 15s	PC → TNC
	calibration terminated	TNC → PC
1110 1ccc	TNC address	PC ↔ TNC

Table 1

Transmission sequence on the radio port

Transmission on the radio port take place according to the following scheme:

1. The PC sends a "TX counter + 1" command to the TNC. The TNC then keys the transmitter (i.e. it activates the PTT line) and increments an internal counter. The PC also increments an internal counter associated with that TNC. This counter serves the PC as a PTT indicator.
2. the PC sends the packet onto the ring
3. as soon as the TX delay value arrives in the TNC, it decides whether enough time has passed already since the transmitter was keyed on. If not, keying up the transmitter is continued until the specified amount of time passed. When the next data byte arrives, the transmission begins, even if the whole packet has not arrived yet.
4. If the TNC receives the "start/end" command and the received checksum was bad, it must send an ABORT sequence on the radio port.
5. As soon as the TNC completes transmitting a packet on the radio port, it sends a "TX counter + 1" command to the PC and decrements its internal counter. Because every packet is preceded by a "TX counter + 1" command from the PC, the TNC always knows how many packets will follow in the current transmission. The transmitter must stay keyed until the last packet is sent.
6. The PC receives the "TX counter + 1" command from the TNC and decrements its associated internal counter. If this counter reaches zero, then the transmission terminates.

The transmission of the channel data packets from the TNC to the PC takes place similarly, with the exception that the PC does not acknowledge a received packet. Every packet the TNC receives on its radio port is sent onto the ring, preceded by an "RX counter + 1" command.

Automatic configuration of the TNC addresses

Automatic configuration of the TNC addresses is achieved by the following simple scheme.

1. the TNC receives an address command (1110 1ccc) and sets its own address to the value contained in the command (the ccc bits).
2. the TNC increments the address contained in the command by one and retransmits the command.

Since every TNC on the ring behaves according to these rules, TNC addresses are allocated sequentially. The command transmitted by the last TNC and received by the PC contains the number of connected TNCs.

KISS versus 6PACK

6PACK provides some advantages over KISS, most notably the ability to transmit real time data and thus the ability to implement the channel access algorithm in the PC.

KISS achieves data transparency by prefixing some data bytes by an escape code. This means that the transmission capacity needed to transmit a data packet depends largely on the contents of the packet. In the worst case, twice the bandwith is needed. With 6PACK, the transmission capacity needed is exactly predictable, if one neglects the real time commands inserted into the data stream.

If one continues the comparison and compares 6PACK rings to KISS token rings, the advantages of 6PACK become even clearer. With a KISS token ring, the response time of the system depends on the loading of the ring. Since the token ring does not transmit DCD and PTT states, each TNC needs to implement its own channel access algorithm, and usually a rather primitive SlotTime/p-Persistence algorithm is used. Also, there is no timing relation between the receipt of a packet over the air and the arrival of the packet in the PC. The PC does not know when the packet is actually sent. (Note by the translator: When the FRACK timer expires and the PC retransmits a frame, the original frame may still wait for transmission, since the channel was busy all the time. Then the retransmission of the frame produces additional unnecessary load on an already congested channel)

This usually leads to the transmission of frames with outdated sequence numbers and to strange effects if one tries to implement a DAMA master on a KISS token ring. With high HDLC bit rates and high load on the ring, the transmitter may be interrupted between packets, which leads to a much increased collision probability.

6PACK provides no mechanisms which would allow the TNC to implement its own channel access. Therefore, the channel access has to be done in the PC. This is a significant advantage, especially on half duplex links.

Short response times, which are needed by, for example, a DAMA slave, cannot be achieved with KISS. The transmitter keying is always delayed by the time it takes to transmit the packet on the asynchronous serial interface.

With 6PACK, real time data and channel data are distinguishable. Therefore, the delay of real time data does not depend on the ring loading, and it can be calculated from the ring bit rate and the number of TNCs on the ring. Each TNC delays the byte by between 10 and 20 bits, which is 260 to 520 µs at 38.4kBit/s. The longest possible delay at 38.4kBit/s is therefore 8 * 520 µs = 4.2 ms.

This means that it does not take more than 4.2 ms to key the transmitter of a particular TNC, or to communicate the DCD state to the PC. This way, even a ring of TNCs may be controlled quite precisely, but the additional delay of a ring may increase the collision probability slightly on a half duplex channel.

A flaw of 6PACK is that the TX delay may be longer than requested. This happens every time a packet is delayed more than the requested TX delay compared to its preceding "TX counter + 1"

command. However, measures should be taken to prevent the peer station from complaining about the too long TX delay.

Last, but not least: Unlike with KISS, the TNC LEDs can be controlled by the host PC directly. They are used just like the LEDs on the RMNC3 card.

Some Remarks from the translator

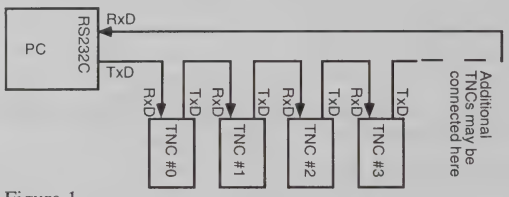


Figure 1

Figure 1 shows the wiring of a TNC ring.

FlexNet allows different channel access algorithms to be used. It may use a modified p-persistence algorithm, DAMA, or OPTIMA in the future. FlexNet allows even multiple data rates and modulation schemes to share one radio channel. For example, combined 1200 Baud AFSK / 9600 Baud FSK user accesses can be found in Europe, since it is often not possible to get different frequencies for both modes, because the 70cm and 23cm bands are very crowded. In this case, a 1200 baud AFSK and a 9600 baud FSK modem are connected to the same transmitter in parallel. Thus the channel access algorithm spans multiple channels. In PC/FlexNet, the channel access for radio channels is done in the L2 kernel. So L1 drivers and 6PACK TNCs must not implement their own channel access algorithm.

TX delay is the time between keying the transmitter and starting to send the first packet. This time allows the radio to start the transmitter. HDLC flags are usually sent during this time. TX delay is specified in 10ms units. FlexNet also monitors the TX delay of the other stations on the channel, and complains if a station uses an excessively long TX delay (more than 100ms in excess to what the transmitter needs). PC/FlexNet L1 drivers and thus 6PACK TNCs should listen for a series of HDLC flags preceding a data packet and report the duration of this sequence in the TX delay byte of the received data packet. Note that only error free reception of the flags should be accepted. In the unlikely event that the hardware does not support the measuring of the TX delay, zero should be returned. Zero may also be returned for every packet other than the first of a particular transmission. If, due to overload on the ring, the required TX delay cannot be met, the sending station must make sure the peer does not complain about the long TX delay. This can be done by inserting an ABORT sequence, for example.

Note that for the sake of fast retransmission of priority commands, hardware FIFOs in the asynchronous serial interface should not be used.

The translator would like to thank Matthias Welwarsky DG2FEF and Gunter Jost DK7WJ for the assistence, and Esther Oswald for correcting this translation.

Related literature:

[1] Karn, Phil KA9Q, and Mike Chepponis K3MC, The KISS TNC: A simple Host-to-TNC communications protocol, 1990

[2] Welwarsky, Matthias DG2FEF, DAMA und OPTIMA, TheFirmware Version 2.7, 1994

Skimming the Layers

A Survey of Software for Logging and Analyzing the Performance of Modern HF Data Communications

Ken Wickwire KB1JY
232 North Rd #17 Bedford MA 01730,
HF Packet/PacTOR/GTOR: KB1JY, KB1JY-2, etc.,
SELCAL: OCRS, APRS: KB1JY,
HF ALE: MB1, MB4, etc.,
VHF Packet: KB1JY@WA1PHY,
Internet E-mail: kwick@mitre.org, etc.[1]

1. Introduction

Many parents have recently announced the rebirth of HF radio through the midwifery of digital signal processing. Newer and older hams have discovered or rediscovered the ionosphere as the place where PacTOR, GTOR, CLOVER and APRS hang out. Other amateurs connected with commercial or government HF are excited about the increasing use of automatic link establishment (ALE), and of data modems with serial- and multi-tone waveforms, forward and reverse (ARQ) error correction, and equalizers. Government HF standards committees are well into the development of sophisticated software for *adaptive* communications at the Data Link and Network layers of HF data-transmission systems.

With this rebirth has come not just pride, but also bafflement and frustration: what modes should I learn and use? Should I spend my money on GTOR, or try my luck with the more expensive CLOVER? How much faster is PacTor II than AX.25? Does it make any sense to use TCP/IP over HF? When amateurs get ALE, how will I decide if it's worth the cost, and cracking my head over the differences between sounding and scanning and calling and linking?

The answer to each of these questions has to do mainly with *performance*: how long will it take to get my file to W1XYZ using PacTOR? Will I clog the 20-meter band with repeated message frames less if I use CLOVER instead of PacTor-II? If I buy an ALE modem, will I really link up with DL4ABC faster than I did when I relied on the *How's DX* predictions in *QST*? Would a serial-tone modem raise my BBS traffic throughput high enough to justify its hefty price?

It would be nice if we could consult a handbook or call an ELMER or dial up a BBS and get a quick, clear and accurate answer to such questions; perhaps we'll be able to some day. To get an accurate answer at the moment (if not necessarily a quick or clear one) takes a combination of theoretical prediction of a system's performance and on-air data to support and qualify it. I'm going to cover some ways to get that data.

Two phenomena have led to the development of the HF digital hardware and software I'll cover: the random, the time-varying ability of the HF channel to support communications, and the arrival of digital signal-processing techniques that can deal with that variation. However, because of the extraordinary difficulty of characterizing the

---

[1] The look of things to come? Sooner or later we may have an Internet-type "Universal Resource Locator" (URL) system for amateur digital communications, and an Internet "Web-server for Hams."

non-stationary randomness of the HF channel, which is affected by fading, multipath, noise and interference, theoretical predictions or on-air experience alone can rarely give convincing answers to questions about performance. While most hams are willing to trust the experts where the derivation of predictions is concerned, both the experts and the rest of us expect in the case of HF that theoretical claims will be backed up by *measurements* of on-air performance.

The purpose of this paper is to describe what it means to "measure HF digital performance" on the air, and to give an overview of what's needed (and available) for making such measurements. It will turn out that in most cases the hardware needed to assess over-the-air performance comes with the system to be assessed: if you have the system (usually a computer, a radio modem and an HF transceiver with an antenna), you have all the hardware you need. A surprising amount of freeware or shareware is also available. I believe that hams using the HF digital modes will increasingly see that they need such software to make sense of the river of bits gushing across their screens; I think the days will soon be over when a few sessions with a microphone or key can convince us that a new digital system is worth buying, learning and using.

For some hams the prospect of setting up new software still brings a shudder. I'll try to show them here that there's welcome news about performance-measuring software: although not everybody can or wants to write it, a great deal of very sophisticated—but increasingly user-friendly—shareware for radio/modem control and performance assessment already exists, with more on the way. In addition, as the number of hams getting into digital HF communications increases, so also does the number who are able and willing to write software that compiler-shy hams can use[2].

This paper surveys of some of that software. I hope the survey will lead more hams to think "digitally" about HF, and encourage developers to write more effective packages for assessing the new modes that are arriving faster than most of us can keep up with them. To cover the software, I've had to provide brief descriptions of the various HF digital modes. The paper may therefore also be useful to those looking for an overview of the latest hardware and protocols available for bit-moving on the shortwave bands.

The paper has five parts. The next (second) part discusses the distinction between link and network assessments and the kinds of measurements needed to do them. The details of how the measurements are made in each case may be further broken down according to the waveform, error-control schemes and communications protocols used (e.g., AX.25, GTOR, TCP/IP, Federal-Standard-1052, etc.), and the specific hardware (TNC, serial-tone modem, etc.) that implements the waveform and protocols.

The third and fourth parts discuss performance assessment for particular cases of widely available digital communications protocols (as implemented in single- and dual-port TNCs, CLOVER-cards, ALE modems, etc.). Part 3 covers links, and Part 4 networks. These parts contain numerous tables and displays of output from assessment programs. In cases where I don't know of any public software for assessing a particular mode's performance, I'll plant some ideas that developers and manufacturers may want to follow up on to fill the gap.

The fifth part is a look into the future: it treats sophisticated data collection schemes developed for the assessment of systems that encrypt data and use separate modems for

---

[2]The desire to see one's name appear on hundreds of computer screens is fortunately also hard for hams to resist.

linking and data transmission. This part also introduces statistical ideas and techniques that may be found useful in understanding and analyzing digital communications data.

I'm sure I've failed to cover somebody's favorite monitoring system. Readers who know about hardware or software I haven't covered should send me info about it at one of the addresses listed above.

Notes. Although the emphasis of this paper is on the HF band, where transmissions experience the most rapidly changing—and thus, most difficult—channel conditions of any radio band, most of the ideas and software discussed should provoke thoughts on the assessment of performance in other bands.

The paper will not cover contest logging software and "traffic monitoring" packages (like Pawel Jalocha's PKTMON1, the Personal Code Explorer™, the MFJ Multi-Reader 1 and Kantronic's GMON and its off-line variants) written mainly to allow third parties to read (sometimes without a modem) the communications of connected stations. Although some of these packages are quite sophisticated, their main purpose is either operator-assessment or listener-amusement, rather than system-performance measurement as such. They are therefore outside the scope of this survey. Also not covered is logging of W0RLI and other BBS usage, since this monitoring concerns mainly "local" operation and is not generally applicable to non-BBS communications.

2. Link versus Network Performance

To keep the discussion simple, I'll define a *link* as a pair of communicating stations with no intervening relays. I define a *network* as a set of three or more stations connected by more than one link.

When we assess **link** performance, we usually have control of both ends of the link and can gather data from at least one of them. This approach, involving only two stations, I'll call *auto-assessment*. It offers the advantages of direct and immediate recording of parameter adjustments and their effects, although recording of requests for retries and numbers of retries is usually difficult. A variant is to use a third station to monitor the performance of two others. I'll call this *third-party assessment*. It offers the contrasting advantages that retries are often easier to record, and that time-tagging and other labeling of monitored packets can often be provided by the third-party's software (for example, in its TNC). In both approaches, we are usually interested only in how the stations on the link cope with channel variations, rather than the details of what other stations are doing and how they affect the link.

In the case of the older, non-adaptive protocols like AX.25, AMTOR and RTTY, we're generally interested in measuring and recording throughput, character-error rates and numbers of repeats. For newer, adaptive protocols like PacTOR, GTOR, PacTOR II and CLOVER, which prescribe *real-time* adjustment of protocol parameters in response to changing communications conditions, we might want to record data rates, interleaver depths, frame-lengths, frames per frame-window, numbers of erroneous frames per frame-window, bit- (or character-) error rates, power levels and so on.

In some cases, the values of these parameters can be recorded or inferred directly from the ASCII screen or file output that appears at one or the other of the link stations. In other words, these values are part of the recorded message traffic. In other cases, the software that implements the communications protocol being assessed determines these values, and either uses them to change frame sizes, etc., or sends them to a modem or

another peripheral device to adjust its performance. In these cases, recording and analyzing the parameter values is (or should be) an option provided by the protocol software, as is the case with several parameters adjusted by the CLOVER system.

In still other cases, the protocol software does not record its parameter changes, and they have to be recorded by other hardware and software that monitors the control and response lines between the system controller (often a PC) and the controlled equipment. An example of such a parameter is the data rate of MIL-STD-188-110A serial-tone modem, which is changed by sending an escape sequence and a speed-change command to the modem.

In some systems, parameter values can only be determined after special hardware modifications of the modem. Examples of this are recording data carrier detect (DCD) indications from older TNCs, and the average Huffman compression ratio in a PacTOR modem. Data rate changes from 100 to 200 bps and back in a PacTOR modem, and between 100, 200 and 300 bps in a Kantronics dual-port KAM with GTOR can be recorded by connected stations themselves using so-called *host-mode* commands (see below), but this may not be possible when monitoring a link between connected stations.

Although manufacturers often provide a means for visual monitoring of data-rate changes (via panel LEDs, etc.), few (with the exception of CLOVER) allow easy recording of these changes. I suggest that the next generation of adaptive modems for HF radio provide this option as a matter of course. Depending on the parameter in question, and whether it is changed by software in the modem or in the PC, parameter changes should be recordable via a dedicated monitoring port on the modem, or as part of the accessible control characters that flow between the controller and the modem.

In assessing **network** performance we are usually interested in numbers like the average throughput of the whole network, the distribution of message delivery times, the sizes of message queues at individual stations, the numbers of relays required to deliver messages and the time it takes a sender to get an acknowledgment of a sent message. The parameters we can adjust to improve network performance, often dynamically in adaptation to changing channel and network conditions, are too numerous to cover completely here. They range from the link-oriented parameters mentioned above, to purely network-oriented ones like message-buffer sizes (a flow-control parameter), routing protocol adjusters and parameters associated with what kind of performance data stations should exchange with each other and how often.

A good example of the latter are the channel quality numbers measured and stored by an ALE modem: if stations exchange channel quality too rarely, the network's ALE modems may try to link on frequencies that no longer support communications; if they send them too often, the network gets bogged down with the traffic generated by channel-quality exchanges, to the detriment of traffic throughput.

The best way to assess network performance is to develop software and hardware that monitors the performance of each station in the network, and make sure that the whole network is under the measurer's control. (An example of such a system is discussed in Section 5.) For amateur networks this is not easy: our networks are big and not under the control of a single group of stations, and our TNCs and multi-mode controllers (excluding CLOVER) were designed for communications rather than data collection. All we can do at the moment is monitor traffic on a local or regional basis and hope that more or less coordinated attempts to tune network parameters (backoff times, packet lengths, etc.) improve performance significantly. I'll cover available software for doing such monitoring below; more is being worked on.

Let's start with approaches to measuring HF link performance and the kind of software needed to do it. Keep in mind that the point of measuring performance is to give operators a valid means for deciding whether some parameter change (data rate, number of repeats before a data-rate change, etc.) improves performance. (Deciding which parameters to change is not always easy, and figuring out why a change leads to better or worse performance is often even less clear, yet both need to be based on performance data.) The presentation will be organized roughly according to the sophistication of each mode's modulation and error control schemes. For a discussion of some performance results on amateur modes following this approach, see the article by Young et al. in the proceedings of the *13th ARRL Digital Communications Conference*.

3. Link Assessment

I'll start at the beginning, where we find modulation schemes with no error control. The two I'll discuss have been used for decades.

3.1. Morse and RTTY Links

In grouping these, I'll assume that the Morse is sent by hand or automatically, but is decoded automatically. (Human decoding of Morse code involves signal processing and error control that are too complicated to be covered by this treatment.) The Morse "signaling set" is ternary (dot, dash and space, with the carrier turned on and off), while the RTTY set is binary (mark and space with FSK or AFSK modulation). In both cases, the signaling alphabets (transmitted character sets) comprise the 26 letters and nine digits plus various punctuation marks and prosigns. The commonly adjustable parameters are the data rate (and perhaps weighting) with Morse, and the data rate (45-300 baud) and shift (170-850 Hz) with RTTY. These can be changed automatically during a connection with appropriate commands sent, for example, by a terminal scripting language like Crosstalk®.

Although I know of no shareware for auto-assessment[3] of Morse and RTTY communications, writing it would be straightforward. A general approach is to send series of messages of fixed length whose content is known at the receiver, and have receiver software compare incoming characters with the expected ones. Since there's no need to assess performance in real time, this requires only terminal software that can capture screen input and store it in a file. (Most modern terminal programs can do this, but a dumb terminal can't.) Care has to be taken (especially with computer-decoded Morse) to avoid unfair penalization of the decoder resulting from a lost character, since these modes offer no error detection or character counting. That is, the shift of character-position caused by the lost character should not normally lead to the counting of multiple errors. The appendix shows the results of a program that compares a received file with a correct file and assesses their differences in various ways.

It's usually a good idea to add time tags to the logged data. This is commonly done at the beginning of each line of received data if the received stream contains line-delimiting control characters. If the file logging is done via a modern programming language like C, Pascal or FORTRAN, this can be effected by calls to the logging computer's clock;

---

[3]Third-party assessment would generally use similar software.

alternatively, some terminal programs' script languages can add time tags to screen-capture data.

Since the options for improving digital communications via Morse and RTTY are relatively limited, let's leave the assessment of these modes and proceed to

3.2. AX.25 ("Packet") Links

Many parameters can be adjusted to affect the performance of an AX.25 packet link over HF (baud rate, packet length, persistence, number of unacknowledged frames outstanding, number of, and time between, retries, slot time, etc.). However, the mode uses only a 16-bit CRC for error detection coupled with an automatic repeat request (ARQ) protocol for error control. This makes it pretty unsuitable for HF, whose channels often require a combination of forward and backward error correction—i.e., an error-correction code *and* an ARQ technique—to achieve throughput of more than a few characters per second. As a compensation for the poor performance, the AX.25 protocol supplies a rich set of commands for monitoring and labeling connected and third-party traffic.

Since received packets are free of errors (unless the PASSALL flag is set, which allows display of packets that fail the CRC), auto-assessment of AX.25 packet performance over a link often amounts to estimating average throughput as a function of the parameters that experimenters think are worth adjusting. To assure that no characters from non-linked stations on the channel corrupt the received stream, the CONLIST switch can be turned on to allow only connections with (and data from) stations listed with the BUDCALLS command[4]. (Remember, however, that "suppressed" stations can still cause collisions.)

The measuring station can turn on the MONITOR, MCON and MSTAMP (or CSTAMP) switches to label packets with date/time stamps. MCOM and MRESP can be turned on to monitor and record AX.25 control packets (<C>, <D>, <UA>, etc.) and AX.25 response packets (<FRMR>, <REJr>, etc.) plus sequence labels for sent or received information packets (<Isr> on a KAM). Aside from software carrier-detect logging, which is not provided by standard TNCs, these commands probably provide all one needs for packet performance logging.

Third-party assessment is likewise aided considerably by the software contained in every TNC (for third-party assessment of network performance, see the Section 4). The MON switch allows various kinds of monitoring of channel traffic[5]. The BUDLIST or SUPLIST switches can be used to restrict monitoring on shared channels, and MSTAMP, MCOM and MRESP can be turned on to time-stamp packets or record control packets. To list monitored stations, the MHEARD command with its various options can be used.

The PASSALL command allows display and capture of frames whose starting and ending flags are recognized, but which fail the AX.25 CRC. Here are a pair of captures on 7.09851 MHz LSB on 20 May 1995, with PASSALL OFF and ON:

---

[4]These are Kantronics commands. In AEA units, similar commands are CFROM, etc.

[5]In some experiments, interference from uncontrolled stations is undesirable, and is avoided by choosing an unused frequency; in others, the effects of interference from other stations are part of the data being monitored, and recording third-party traffic on a shared frequency is acceptable.

PASSALL OFF:

```
WA2SPL>K1RQG/H:Power Module VHF ?
R:950518/1824 5176@WA2SPL.VT.USA.NA
R:950517/2039 1@ON7RC.#BR
WA2SPL>K1RQG/H:rig.
First I had irregular transmitions on VHF, and finaly found the final Hybri
WA2SPL>K1RQG/H:rig.
First I had irregular transmitions on VHF, and finaly found the final Hybri
WA2SPL>K1RQG/H:d power module to be the reason.
Can someone tell me the characteristic differen
WA2SPL>K1RQG/H:d power module to be the reason.
```

Here are the same two stations a few moments later with **PASSALL ON:**

```
Hi and many thanks for read this.
#48    ú! & fOom m
WA2SPL>K1RQG/H:ay 16th to 31th at 24:00 UTC the special*station
EG1RD  will be on the air. This
WA2SPL>K1RQG/H:ay 16th to 31th at 24:00 UTC the special station

Dried herbs: Keep in cool, ark place. They generbloy bffin to lo
WA2SPL>K1RQG/H:òetwose not5d.
Dried herbs: Keep in cool, ark place. They genUq=ÿfiÍ@èñô·πÅ-ΩÅ∝Ω
u˜°^b$¢é@‡ÆÇd¶†òaſ●se potency
within 6 months. Crush in fingers to check for aroma. May be refrige
WA2SPL>K1RQG/H:se potency
 within 6 months. Crush in fingers to check for aroma. May be¨ºefrige
WA2SPL>K1RQG/H:rated if you have room.
```

Notice in both cases the repeated frames, and in the second case, the frames (with one or more nonsense characters—here interpreted by a Macintosh) that failed the CRC.

Recent graphical user interface (GUI) terminal programs (and several monitoring and other packages now in the works) use the KISS interface to talk to TNCs, which usually allows faster access to TNC functions and data (compare this with use of the host mode interface, which has a similar purpose). *Savant* for the Mac by Jim Van Peursem (KE0PH) is an example of such a program. Savant's user screen displays, in real time, the number of MAXFRAME packets that have not yet been acknowledged, the number outstanding, the number of retries and the round-trip time. Unfortunately, the current version apparently provides no way to record these statistics.

3.3. AMTOR Links

AMTOR (and its parent SITOR) is the first widely used system with forward error correction (FEC or "Mode B," using simple two-fold character-repetition), error detection (based on a check that each seven-bit character has four ones and three zeros) and ARQ (the chirping "Mode A"). The combination of forward and backward error correction, typical of more modern systems, is not provided by the AMTOR protocol.

In the FEC mode, which sounds like RTTY, detection of just one erroneous character in a repeated pair causes the "correct" character to be printed; if both characters are declared erroneous, or both are "correct," but don't match, a "missing character" symbol is printed. (In some TNCs this symbol can be specified, which may make analysis of character errors by a parsing program easier.) This allows for simple (usually off-line) comparison of received characters with sent ones for auto assessment. Throughput and estimated character error rate are the usual measures of performance, and programmed

comparison at the receiver of stored copies of sent messages the standard means of assessment.

Third-party assessment of stations communicating in either FEC or ARQ mode is possible. (Some multimode TNCs monitor one mode as a default. The Kantronics KAM "LAMTOR" command allows automatic reception of either mode.) Monitoring at a well-placed location of FEC transmissions (whereby the monitor decodes the repeated characters) or ARQ exchanges might allow a useful independent assessment of which is more effective, since the transmitting and receiving stations have no easy way of automatically recording the number of repeated characters in the ARQ mode. On the other hand, because it is not in dialog with the sender, the third party monitor may entirely miss characters repeated in the ARQ mode.

Auto- and third-party assessment can be aided in the AMTOR, PacTOR and GTOR modes by turning on the TRACE command. This allows display and recording (in HEX) of full header information in addition to information content; in particular, ACKs and NACKs can be monitored with this command set to ON.

Since few parameters can be adjusted to improve AMTOR communications (among them, the mode (FEC or ARQ) and in some TNCs, the delay between receipt of a character triplet in the ARQ mode and its ACK/NACK), the system's further development for HF digital communications is limited. Let's turn therefore to the first HF system to offer *automatic* adaptation to shortwave channel conditions.

3.4. PacTOR Links

PacTor has, like AMTOR, both FEC and ARQ modes, but differs from it in providing automatic adjustment of data rate (100 or 200 baud) and Huffman data compression. PacTOR also allows a variation of conventional ARQ called "memory ARQ," in which settable numbers of repeated but erroneous frames are saved in an attempt to reconstruct a single correct version of the frame. The version of PacTor used by most hams, and treated here, uses binary FSK, like packet. German hams have recently begun marketing PacTOR II, which uses binary and higher-order phase-shift keying (PSK), and differs in some other respects from ordinary PacTOR.

Several parameters can be adjusted to regulate how PacTOR adapts its data rate and ARQ scheme to channel conditions (the data rate can also be set manually). In a KAM (other implementations are similar) these parameters are

- the number of consecutive erroneous packets that cause the data rate to be automatically lowered to 100 baud,
- the number of consecutive error-free packets that cause the data rate to be automatically raised to 200 baud,
- the allowable number of *unsuccessful* attempts to increase the baud rate before it is raised automatically only by the previous criterion,
- the baud rate, or the choice of automatic baud-rate selection,
- the number of repetitions of each frame in the FEC mode,
- the number of link attempts or consecutive erroneous frames before time-out and
- the number of erroneous frames stored and used to construct an error-free frame with memory ARQ.

Although manufacturers advise that data rates be set automatically, the fact that many parameters (including data rate) can be set by the user gives wide scope for experiments

with adaptive link control. For those who don't want to tackle serial I/O programming on a PC or Macintosh (not generally for the faint-hearted), powerful scripting languages like Crosstalk® can be used to implement adaptive control of PacTOR communications.

Auto-assessment of PacTOR performance can be achieved by calculating throughput in the usual way (with calls to the processor clock), by comparing received with stored text in the FEC mode, and by displaying and recording (through screen-capture) the supervisory information exchanged by PacTOR stations. (Some—and perhaps all—current implementations of PacTOR fail to include this supervisory information, and the command to display it is a mere place holder.)

Third-party assessment of both FEC and ARQ communications is possible, and may be aided in some experiments by displaying (when possible) exchanged supervisory information at the monitoring station.

Another monitoring command that's useful for PacTOR assessment is TRACE, which can also be used in other HF modes. The TRACE command allows display and capture (in HEX) of all monitored frames, with frame data-contents also displayed in ASCII.

Here's a capture of a PacTOR exchange on 14.07666 MHz LSB, on 20 May 1995, monitored with PTLISTEN, first with TRACE OFF, then with it ON, a few moments later:

TRACE OFF:

```
Software: [33mXP that LL Joachim and get a copy of XPCOM... much
much better program than the poor LL... XPCOM and QE
BIU Joachim..
is PK-232 for the KAM...
702 S. Ashbrook
On this side of the Atlantic 9 out every using this
program... You should get a copy from  look at BOOTUPS it needs
registration nne ctip? How does the si F5VAU de YV1AQE..
My beam is stuck pointing to the USA Joachim, so your signal
not strong sounds clean... As S-3
F5VAU de YV1AI
I hier this evening on 40 m decribing the signal aOk willy, the
NOTHING wrong our and sharp and with other ststions reseption 40
hi Have a nice
Joachim.. And x
YV1AQE
```

TRACE ON:

```
[AA2020202020202020202020202020202020BA0D0128E5]
[AA2020202020202020202020202020202020BA0D0128E5]
[AA2020202020202020202020202020202020BA0D0128E5]
[5546355641550F0F046355641551E]<C:F5VAU>

[AA317976316171650DCDCDCDCDCDCDCDCDCDCDCD0105A4]yv1aqe
[AACDCDCDCDCDCDCDCDCDCDCDCDCDBC0D0A1B5B33316D032FE8]
[5520202020B1B1B1B1B1B1DC2020B1B1B1B1B1B1DC005384] ±±±±±‹  ±±±±±‹
[5520202020B1B1B1B1B1B1DC2020B1B1B1B1B1B1DC005384]
[5520202020B1B1B1B1B1B1DC2020B1B1B1B1B1B1DC005384]
[AA0D0A1B5B33316D2020202020DFDFDFB1B1DB202001DE57]
[31m     ff1fl±±¤
[5520DFDFDFB1B1DB1B5B33356D20205468616E6B7302EB68] ff1fl±±¤[35m  Thanks
[5520DFDFDFB1B1DB1B5B33356D20205468616E6B7302EB68]
[5520DFDFDFB1B1DB1B5B33356D20205468616E6B7302EB68]
```

```
[AA0D260919597A1A5CBA00E2867DA8A50F8F87C7C307DD2C] Joachim, for the very n
[556963650D0A1B5B33316D20202020202020B1B1B1004210]ice
[31m       ±±±
[556963650D0A1B5B33316D20202020202020B1B1B1004210]
[AAB1DB2020B1B1B1B1B1DB1B5B33366D2020504143011134F]±¤  ±±±±±¤[36m  PAC
[552F2806F51A91829F443757BA13AB66D887EDC3E306F80C]TOR qso from Lorraine
[AA5B33316D20202020202020202020B1B1DBDF202020DF03C3E4][31m        ±±¤fl   fl
[AA42E3520B12B8588D48924E18F6614BB178783C3C05F9DE]an do it again soon.
[AA42E3520B12B8588D48924E18F6614BB178783C3C05F9DE]
[5533316D20202020202020202020B1B1DB2020B1B1B1B102EE9D]31m        ±±¤  ±±±±
[AAB1B1DB1B5B33326D202054616B6520636172652C03D1A4]±±¤[32m  Take care,
[5509E50D4116C4C9028212B8589D300F8F87C7C3E3040DFF] and please call again.
[55DFDFDFDFDF0D0A1B5B313B33373B34306DC9CDCD02E487]ffffffff[1;37;40m
[55DFDFDFDFDF0D0A1B5B313B33373B34306DC9CDCD02E487]
[55CDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCDCD002256]
[AACDCDCDCDCDCDCDCDCDCDCDCDCDCDCDBB0D0ABA202001A9D0]
[AACDCDCDCDCDCDCDCDCDCDCDCDCDCDCDBB0D0ABA202001A9D0]
[555514433114D32B7142160B93A68D61F5E8E1F1F006A56D]*** Maracaibo, Venez
[AADF9008F5BDA50F902869DFCA228AA1188A79783C07FDA8]uela - South America ***
[555B306D0D0A4164696F73201E1E1E1E1E1E1E1E1E0077DD][0m Adios
[AA4A6F616368696D2E2E2E201E1E1E1E1E1E1E1E1E01CF18]Joachim...
[551E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E1E022476]
[5514EB1814285F1F08F8E8550156F1300B901E1E0F045F2C]YV1AQE terminating with
[AA33A88E4139F864C2BC674980D0428BE268A3DBE0051431]F5VAU.. SK at 22:36 U
[5554432E2E0D201E1E1E1E1E1E1E1E1E1E1E1E1E1E02A5BA]TC..
[5554432E2E0D201E1E1E1E1E1E1E1E1E1E1E1E1E1E02A5BA]
[AA1E1E1E1E1E333303303C33330F3C3033331E1E1E832D70]<D:F5VAU33>
[AA1E1E1E1E1E333303303C33330F3C3033331E1E1E832D70]
[AA1E1E1E1E1E333303303C33330F3C3033331E1E1E832D70]
```

Automatic recording of data-rate changes when PacTOR is in its automatic baud-rate-adjusting mode is not possible when the modem is controlled from a conventional terminal program. This would require (on a KAM, for example) monitoring and recording the state of the LED that displays baud-rate switches. (The KAM and other TNCs can, however, be interrogated via their serial ports for data rate, PTT status, sending and receiving status, FEC vs. ARQ mode, etc., when they're in the *host mode*. KGOLD, XPKAM and other host mode applications allow such interrogation.)

3.5. GTOR Links

GTOR also has FEC and ARQ modes, and provides automatic adjustment of data rate from among 100, 200 or 300 baud. The (pure) GTOR FEC mode is the same as the AMTOR FEC mode (repetition of characters) and is used for broadcasting. For its ARQ mode, GTOR applies Golay forward error-correction and block interleaving to transmitted data. At the receiver, the data are de-interleaved and checked for errors with a CRC. If errors are found, retransmission of the data is requested. The receiver analyzes the retransmitted data for errors and applies Golay error-correction to them if necessary. If this doesn't correct all errors (it often does), another retransmission is requested. This is an example of what's called *adaptive ARQ*.

GTOR can apply three kinds of data compression to lower the number of bits used to send characters, and can be set to tolerate a small number of errors in frame acknowledgments.

As with PacTOR, several parameters can be adjusted to regulate how GTOR adapts its data rate and ARQ scheme to the channel (the data rate can also be set manually). In a KAM these parameters are

• the number of consecutive erroneous packets that cause the data rate to be automatically lowered to 200 or 100 baud,
• the number of consecutive error-free packets that cause the data rate to be automatically raised to 300 baud (no automatic increases to 200 baud occur),
• the allowable number of *unsuccessful* attempts to increase the baud rate before it is raised automatically only by the previous criterion,
• the baud rate, or the choice of automatic baud-rate selection,
• the allowable number of erroneous bits in an ACK ("fuzzy" ACKs) and
• the number of link attempts or consecutive erroneous frames before time-out.

The fact that many parameters (including data rate) can be set by the user also allows adaptive link control of GTOR communications. Scripting languages are again an easy way to implement adaptive control of GTOR links.

Auto-assessment of GTOR performance can be achieved by computing throughput and by interrogating the serial port for data rate, PTT status, sending and receiving status, etc. Host-mode programs can apparently do this already.

Third-party assessment of GTOR ARQ communications is not possible with a standard terminal program because the KAM's processor is not fast enough to distinguish and decode packets in real time[6]. Kantronics has produced a program called GMON that uses the processing power of a 286- or faster PC to assist in monitoring the traffic between connected stations. This traffic is delivered to GMON in the form of "scanned" samples of data from the KAM. On slower PCs recorded scanned samples can be monitored off-line with a program called GOFF. GMON does not apply error correction to received data, and allows in that sense comparison of error-corrected and non-corrected transmissions. A third program, GMONITOR, allows off-line analysis of error-corrected data.

3.6. CLOVER Links

CLOVER is the most advanced adaptive modem sold on the amateur market. The modem modulates and demodulates successive pulses at four pulse frequencies with any one of seven waveforms, including BPSK, QPSK, 8PSK, 16PSK and combinations of 8PSK with two amplitudes and 16PSK with four amplitudes. (The latter are called *quadrature amplitude modulation*, or QAM.) The four-pulse waveforms occupy no more than 500 Hz of bandwidth.

CLOVER provides—like AMTOR and PacTOR—an FEC mode and an ARQ mode. In the ARQ mode—the main adaptive CLOVER mode—, the modem automatically chooses the "best" waveform in accordance with measured channel conditions. Since different bit rates are associated with different waveforms, the modem has an adaptive data rate. Its channel rate (data bits plus overhead bits) can be set (normally automatically) between 31.25 and 750 bps (with data rates up to about 500 bps). The CLOVER *symbol rate* is constant, and is 31.25 baud (symbols per second). Using signal-to-noise ratio measurements at their receivers, two CLOVER stations in the ARQ mode can also adjust their output powers for most efficient channel usage.

---

[6]The v8.0 KAM software update is said to remedy this shortcoming.

Forward error correction is provided by a Reed-Solomon code—a powerful, non-binary (symbol) block code that can correct burst errors. Codeword lengths of 17, 51, 85 and 255 bits may be chosen. The code's rate (information bits/total bits) is adjustable to 0.6, 0.75, 0.90 and 1.00, with corresponding capabilities of correcting codewords containing 20, 12, 5 and 0% incorrect bits.

Backward error correction in very poor channels is provided by an ARQ approach that uses the error-detecting capability of a check-sum and *selective repeats*. This efficient, but software-intensive approach to ARQ involves requests for repetition of only the erroneous frames in a "frame-window" (set of transmitted frames)[7].

The modem can collect (and pass on for display and processing) channel-quality and labeling information in the ARQ mode as follows :

- time tag,
- sender's or receiver's callsign,
- modulation format,
- FEC coding rate,
- throughput (bytes per second),
- SNR (dB),
- frequency offset (Hz),
- phase dispersion,
- used Reed-Solomon FEC capacity and
- transmitted power.

(In the FEC mode, the receiver's callsign is not recorded, and the coding rate and transmitted power are fixed.) The information listed can also be recorded automatically to a file. The recording format is the comma-delimited line of strings

[time],[call],[modul.],[code rate],[data rate],[SNR],[freq. offset],[ph. disp.],[FEC cap.],[pwr]<CR><LF>

This facility meets most requirements for auto-assessment. Additional off-line analysis of performance can be performed with a spreadsheet or graphing program, or with other parsing and statistical software that processes the time-tagged statistics files.

If only SNR and phase distortion data are required, two CLOVER stations can put themselves in the manual ARQ mode, choose BPSK modulation and send no message data. The stations will then function as a "bi-directional oblique-incidence sounder-system," exchanging only data on signal quality. (See the discussion of ALE systems below for other low-cost HF channel-sounding possibilities.)

Third-party assessment of CLOVER is possible at a CLOVER-equipped listening station. Both FEC and ARQ transmissions may be monitored—provided that the listener has correctly received the appropriate connection blocks[8]—, and Reed-Solomon error-correction of monitored data blocks can be performed. However, because of the tight coordination of sending and receiving stations required for the selective-repeat ARQ mode, error correction via repeated transmissions is not possible at third party stations.

---

[7] AMTOR and PacTOR use so-called "Stop-and-wait" ARQ, and Packet uses so-called "Go-back-N" ARQ. Both approaches generally lead to lower throughput than Selective-Repeat ARQ.

[8] If the connection block (which contains the caller's callsign and announces the waveform) is missed, the listener has to wait for the next connection block, which may take 20-30 seconds.

In many respects, the CLOVER statistics-recording scheme is a model for what future adaptive HF digital communications systems should provide. The data collected by CLOVER offer fascinating opportunities for experimentation and performance improvement. I hope that other vendors and developers will agree, and include corresponding recording capabilities as options in their own products.

3.7. TCP/IP Links

TCP/IP communications over HF are possible using various versions of NOS for PCs or of NET/Mac for the Mac. The software uses a TNC running in the KISS mode. TCP/IP communications are very slow because of the 300-baud HF data-rate restriction, the overhead caused by network- and transport-layer header information and an error-control scheme (ARQ but no FEC) that is not tailored to HF radio operations.

In addition to the standard IP-controlled communications at the network level, AX.25 and NET/ROM connections can also be established from within amateur TCP/IP applications. In each case, TCP and IP packets are actually imbedded in AX.25 or NET/ROM frames for transmission over TCP/IP links or networks. The slowness of TCP/IP over HF is a pity since amateur implementations of TCP/IP offer, with their Internet relatives[9], telnet and FTP sessions, binary file transfers, pinging, SMTP, POP mail and the provision of extensive performance monitoring.

The parameters that can be adjusted to tune TCP/IP performance are too numerous to list here in their entirety. Among them are: various AX.25 parameters (paclen, etc.), NET/ROM timing parameters, maximum number of hops before a packet is discarded, maximum segment (read frame) size, maximum frame-window size and virtual circuit (link and IP level CRCs and ACKs) vs. datagram (IP level CRCs and ACKs only) mode. For HF operation many of these parameters should have much different values than they have for wire or line-of-sight VHF or UHF operations.

Numerous statistics on connection status and communications performance are provided by TCP/IP. Among the most interesting and useful are the initial and subsequent round-trip ping times, which measure the time (usually in milliseconds) it takes a packet of settable length to reach a called station and be returned. Here are the results of pairs of pings (sent at 1200 bps) over a VHF/UHF link via the gateway WA1PHY to W1IMM-2 (40 miles away) and over a VHF link to WA1PHY itself (three miles away):

```
Ockers> ping w1imm-2
Ockers> 44.56.8.102: echo reply id 0 seq 2052, 11800 ms
Ockers> ping w1imm-2
Ockers> 44.56.8.102: echo reply id 0 seq 2206, 6000 ms
Ockers> ping wa1phy
Ockers> 44.56.4.1: echo reply id 0 seq 2403, 2800 ms
Ockers> ping wa1phy
Ockers> 44.56.4.1: echo reply id 0 seq 2466, 5100 ms
```

The first ping to W1IMM-2 took longer than the second because a suitable route to W1IMM-2 had first to be found; the second ping to WA1PHY took longer than the first probably because WA1PHY was busy during the second. Suitably calibrated ping times

[9] I'm told that we can expect within the year an AX.25 implemention that uses the standard TCP driver of a popular PC operating system. This would allow hams to use several TCP performance-monitoring programs to monitor packet-radio communications.

might be useful over HF links as an aid in propagation studies (perhaps of surfacewave vs. skywave modes).

The KA9Q TCP/IP implementations used in ham radio also provide statistics on AX.25, NET/ROM, IP and TCP protocol connections. Here are examples of queries (during a cross-connected VHF/UHF telnet session between KB1JY and W1IMM-2 as to the status of AX.25, NET/ROM, IP and TCP connections:

```
Ockers> ax status
    &AXB IF   Snd-Q   Rcv-Q   Remote    State
  2379cc ax0  0       0      WA1PHY-6  Connected
Ockers> nr status
Interface  SndQ  RcvB  NumReceived  CSumErrors
Ockers> ip status
IP: total 26 runt 0 len err 0 vers err 0 chksum err 0 badproto 0
ICMP: chksum err 0 no space 0 icmp 0 bdcsts 0
type  rcvd  sent
Ockers> tcp status
conout 1 conin 0 reset out 0 runt 0 chksum err 0 bdcsts 0
 #    &TCB Rcv-Q Snd-Q  Local socket          Remote socket        State
 0  20f390     0     0  kb1jy:9               0.0.0.0:0            Listen (S)
 4  20f14c     0     0  kb1jy:21              0.0.0.0:0            Listen (S)
 6  20f088     0     0  kb1jy:23              0.0.0.0:0            Listen (S)
 9  20cdc8     0     0  kb1jy:1024            w1imm-2:23           Established
    20f2d0     0     0  kb1jy:7               0.0.0.0:0            Listen (S)
 a  20f458     0     0  kb1jy:79              0.0.0.0:0            Listen (S)
10  20f20c     0     0  kb1jy:25              0.0.0.0:0            Listen (S)
```

All of these (here either empty or rather uninteresting) statistics might be useful in assessment of HF communications that use TCP/IP-frames encapsulated in AX.25 frames.

NOS and NET/Mac also have a TRACE (and record) capability that can be used for third-party assessment of AX.25 (packet), NET/ROM and TCP/IP communications over HF (and VHF). The TCP/IP TRACE command can be tailored to display output or input headers and the information content of frames in ASCII or HEX or both. Here's an example of some packet traffic on 14.103 MHz monitored using the "trace headers only" version (TRACE ax0 011):

```
Mar 19 18:24:50 ax0 recv:
AX25: W9YHY-15->N8HT-7 RR(P) NR=7
Mar 19 18:24:52 ax0 recv:
AX25: W9YHY-15->N8HT-7 I NR=7 NS=0 pid=Text
Mar 19 18:26:27 ax0 recv:
AX25: W9YHY-15->N8HT-7 RR(F) NR=0
Mar 19 18:26:34 ax0 recv:
AX25: KD4NDH-13->KD1MT SABM(P)
Mar 19 18:26:40 ax0 recv:
AX25: KD4NDH-13->KD1MT SABM(P)
Mar 19 18:26:44 ax0 recv:
```

Here's some traffic on 7.098 MHz with both headers and information content displayed (TRACE ax0 111):

```
Mar 20 21:12:36 ax0 recv:
AX25: N1GMU-1->NODES UI pid=NET/ROM
NET/ROM Routing: WVT
```

```
    N1GMU  BBSGMU     N1GMU-1   30
    VE2RM  WQC        VE2RM     10
  W2UXC-1  PLB        W2UXC-1   10
 WA2SPL-1  NWVT       WA2SPL-1  10
                        .
                        .
                        .
Mar 20 21:13:42 ax0 recv:
AX25: N1GMU->K1RQG I(P) NR=3 NS=4 pid=Text
0000  .TX.USA..the computer.  The ''Y'' cable
Mar 20 21:13:44 ax0 recv:
AX25: N1GMU->K1RQG I NR=3 NS=5 pid=Text
0000  is NOT needed..What you do need is
```

Note that these trace data contain a NET/ROM routing table and some ASCII text.

Here, for comparison, is a screen capture of some HF packet communications on 21.097 MHz with the TNC commands MON, MCOM, MRESP, MSTAMP and PASSALLPID ON:

```
W4DPH>VE1AIC/H [20-03-94  19:01:28]: <<C>>:
W4DPH>VE1AIC/H [20-03-94  19:01:39]: <<C>>:
W4DPH>VE1AIC/H [20-03-94  19:02:28]: <<rr0>>:
W4DPH>VE1AIC/H [20-03-94  19:02:34]: <<rr0>>:
W4DPH>VE1AIC/H [20-03-94  19:02:45]: <<rr0>>:
W4DPH>VE1AIC/H [20-03-94  19:03:37]: <<rr0>>:
N7JOR>K4OLC-3/H [20-03-94  19:13:00]: <<UA>>:
N7JOR>K4OLC-3/H [20-03-94  19:13:02]: <<I00>>:{F0}HELLO, NAME HERE IS
*EDDIE*
```

Since TCP/IP trace data can be written to files, further off-line analysis of trace data is easy. There is a TRACEONLY version of the TCP/IP trace command that allows recording of only TCP/IP packets to or from a particular station. This is the equivalent of the packet BUDLIST ON command (with one station in the BUDCALLS list) for third-party assessment of TCP/IP performance.

The NOS AT Command

Recent versions of the NOS TCP/IP application for the PC have a command called AT (for automatic timing) that allows one to schedule NOS actions like PING, FINGER and STATUS. This useful command allows TCP/IPers to design propagation and communications experiments in which one station automatically interacts with one or more other ones at regular intervals. The same command can be used at sending or receiving nodes to query NOS on the status of attached ports and the statistics of communications at the link, network or transport level (AX.25, IP, NET/ROM or TCP status). If the results of AT commands can be recorded in some way (session recording, screen buffer capture or recompiling NOS to send AT output to a file), one could avoid the use of more complicated scripts or lower-level code in performance assessment.

For example, to ping K5XYZ every five minutes and record the results, one could send the command

AT NOW+0005 "PING K5XYZ +"

followed by an appropriate recording command. (Pinging can, of course, be scheduled with the ping command itself.) In a two-way communications experiment involving scheduled calls, the called station might record the results of the command

which queries NOS every 20 minutes on the status of communications at the TCP layer[10].

3.8. Federal-Standard-1045 (ALE) Links

Automatic link establishment (ALE) may be the most powerful and interesting development in HF since SSB. The protocols for it were standardized in the late 80s, and ALE equipment has been commercially and widely available for about six years. Since ALE requires complicated control of receiver scanning, an ALE modem is usually mated to an HF transceiver as a built-in accessory, but stand-alone models also exist. Prices have been slowly but steadily falling, and there's a good chance that hams will soon be able to afford and use ALE on the HF bands.

The ALE waveform is 8-ary phase-continuous FSK. The mode's error-control was designed for very high reliability at the expense of throughput—a normally inescapable tradeoff in the case of HF. Error-control is provided by a combination of Golay forward error-correction coding, interleaving and three-fold bit repetition[11]. At the receiver a majority vote on the repeated bits, de-interleaving and Golay decoding usually assure that there are no errors in ALE words delivered to the communications terminal.

In an ALE network, idle stations are usually scanning a programmed set of between five and ten frequencies, waiting for calls. At regular intervals during the day an agreed-upon subset of stations broadcast short sounding transmissions on each of the frequencies. Any scanning station that hears a sound on a frequency automatically records the address of the sender and the quality of the signal. A special sounding variation called a link quality assessment (LQA) exchange allows a particular pair of stations to measure and exchange on-air quality measurements of the HF channels between them.

These stored channel qualities, if broadcast (or exchanged) by the right stations and recorded often enough (three or four times a day is usually enough), allow network stations to choose the best frequency for communication with any other stations at any time. While some discipline (or restrictions) will have to be imposed on ham sounding if a large number of us use ALE, the technique clearly offers exciting possibilities for great improvement of amateur communications over HF.

When a station wants to call another, it automatically looks through its channel-quality memory for the frequency with the highest quality. It then calls the desired station on that frequency. If the called station hears the call and recognizes its address, it answers. If he hears the answer, the caller completes the "three-way handshake" with an acknowledgment of the called station's answer. The two stations are then linked, and may communicate using voice or data (Packet, PacTOR, TCP/IP, etc.). Several more complicated kinds of calls to multiple stations (in nets that expect to be called, or "on-the-fly"), to stations in other networks (ANYCALLS, ALLCALLS), and to stations with common address characters (WILDCARD calls) can also be made.

The software built into an ALE modem (but unfortunately not yet standardized across manufacturers) allows extensive performance monitoring and channel assessment. All

---

[10]AT-scheduled sessions are turned off with an AT K N command, which "kills" session N.

[11]An optional orderwire (text-transmission) mode adds an ARQ technique for further error control.

implementations allow automatic, scheduled sounds, storage of LQA scores and recall of the addresses of called stations with their associated LQA scores. Commands to rank all channels used by a sounding or communicating station allow regular comparison of channel qualities.

Here are some captured two-way channel rankings taken with a 125-watt, ALE-equipped, Harris RF5000 transceiver for channels between Bedford, Mass., near Boston, and NFK in Norfolk, Mass. (40 miles), MT2 in Reston, Va. (350 miles), and MX1 in Ft. Wayne, In. (800 miles). The remote radios also transmitted about 125 watts. The recorded "scores" are combinations of the "measured" SNR (received at Bedford) and Bedford's SNR (received at the distant station and sent as part of an LQA exchange back to Bedford). The largest possible score is 120, and a received SNR of 31 in the case of MT2 and MX1 indicate that MT2's and MX1's data—although compliant with the ALE standard—did not conform to the format used by the RF5000 for complete display of two-way measurements. This is a common problem resulting from the lack of standardization of measurement display and score calculation.

```
12/15/94
4:16:00 PM (GMT)
RANK  NFK
CHAN: 08  SCORE: 015    MEASURED SNR 10  RECEIVED SNR 30
CHAN: 07  SCORE: 007    MEASURED SNR 07  RECEIVED SNR 17
CHAN: 06  SCORE: 003    MEASURED SNR 15  RECEIVED SNR 26
CHAN: 05  SCORE: 001    MEASURED SNR 10  RECEIVED SNR 13

12/15/94
4:16:37 PM
RANK  MT2
CHAN: 07  SCORE: 095    MEASURED SNR 30  RECEIVED SNR 31
CHAN: 09  SCORE: 040    MEASURED SNR 14  RECEIVED SNR 31
CHAN: 08  SCORE: 039    MEASURED SNR 14  RECEIVED SNR 31

12/27/94
20:22:44 PM
RANK  MX1
CHAN: 10  SCORE: 085    MEASURED SNR 21  RECEIVED SNR 31
CHAN: 09  SCORE: 075    MEASURED SNR 15  RECEIVED SNR 31
CHAN: 08  SCORE: 071    MEASURED SNR 14  RECEIVED SNR 31
CHAN: 11  SCORE: 059    MEASURED SNR 07  RECEIVED SNR 31
CHAN: 07  SCORE: 057    MEASURED SNR 08  RECEIVED SNR 31
```

Auto-assessment of ALE-assisted communications or channel quality can be accomplished using a script language or a lower-level language like C to send commands and queries to the modem. Third-party assessment requires an ALE modem and receiver for detecting and decoding ALE transmissions. The third party can record sounds, or carry out LQA exchanges with particular stations in the scanning mode. Because of the brief time a third-party listener spends on a channel before returning to scan if he does not recognize his own address in a call, third-part monitoring (especially of message traffic) is normally performed only on fixed channels (i.e., in the ALE single-channel mode).

3.9. Federal-Standard-1052 (HF Packet Radio via Serial-tone modem) Links

Federal standard-1052 is still in development, but should be published within the year. The packet protocol part of the standard assumes that a suitable communications frequency has already been found, and prescribes more or less continuous "negotiation" between pairs of stations on that frequency that is similar to what happens with the

CLOVER protocol. The negotiation starts with a linking exchange and is maintained with negotiation packets called "heralds" and "herald-ACKs." The protocol's error-control is provided by selective-repeat ARQ, which is the most efficient of the non-hybrid ARQ techniques.

Although 1052 does not prescribe a specific ALE technique, it is often implemented in systems that also use Federal-Standard ALE. No data modem is prescribed by the 1052 packet protocol either, but it is generally implemented with a Military-Standard-188-110A serial-tone modem. This modem (currently too expensive for most hams) uses convolutional forward-error-correction coding, a variable-length interleaver, a narrowband (<< 3 kHz) interference excisor and a channel equalizer that uses training frames interspersed with data frames to compensate for waveform distortion caused by HF multipath. The combination of the serial-tone modem with 1052's ARQ protocol probably provides the most effective data transmission available for the HF channel[12].

Since 1052 is still in development, assessment of its performance is beyond this paper's scope. On the other hand, the serial-tone modem that usually does the dirty work for the protocol has been around for nearly a decade. The modem has a fairly simple command set (basically TEST, OPER, set INTERLEAVER and set DATA RATE). A higher-level protocol like 1052 uses these modem-commands to adapt itself to channel changes.

The serial-tone modem has (like CLOVER) the ability to measure and send to Data Terminal Equipment (PC) almost continuous measurements of post-processing received signal-to-noise ratio (i.e., after interference excision and equalizing). These measurements, produced at the rate of two or three per second, can be recorded by a terminal program with screen capture, and are a part of auto-assessment of performance. Although they have to be analyzed with some care because they represent SNRs *after* signal processing, these data provide an easy and useful means of studying HF fading and interference. Here's an example of some control-line output from a serial-tone modem. (Message data output can also be monitored.) The control data were gathered during recent daytime reception of on-air message data at 4.5 MHz from a station about 250 miles away. The raw ASCII control data (not shown here) have already been processed off-line by a parsing program that has removed blank lines and labeled status flags ("ready bit on," etc.).

```
Input file is [02012053.IPM].
Parsed output:

20:58:18 [14;6f[KOPER
21:06:14 [16;9f[K    [19;9f[K0100   ready bit on
21:06:24 [16;9f[K -- [19;9f[K0100   busy channel on
21:06:25 [16;9f[K -- [19;9f[K0100
21:06:25 [16;9f[K -- [19;9f[K0100
21:06:26 [16;9f[K -- [19;9f[K0100
21:06:26 [16;9f[K -- [19;9f[K0100
21:06:26 [16;9f[K  09[19;9f[K0100
21:06:27 [16;9f[K  09[19;9f[K0100
21:06:27 [16;9f[K  09[19;9f[K0100
21:06:28 [16;9f[K  09[19;9f[K0100
21:06:28 [16;9f[K  10[19;9f[K0100
21:06:28 [16;9f[K  10[19;9f[K0100
21:06:29 [16;9f[K  10[19;9f[K0100
21:06:29 [16;9f[K  10[19;9f[K0100
```

[12]HF Data-link protocols developed recently by the NATO SHAPE Technical Centre in Holland, and by the airlines for automatic position reporting of ocean-flying passanger aircraft, use serial-tone modems to produce similar performance.

```
21:06:29 [16;9f[K  10[19;9f[K0101   synch bit on
21:06:30 [16;9f[K  11[19;9f[K0101
21:06:30 [16;9f[K  11[19;9f[K0101
21:06:31 [16;9f[K  11[19;9f[K0101
21:06:31 [16;9f[K  11[19;9f[K0101
21:06:31 [16;9f[K  11[19;9f[K0101
21:06:32 [16;9f[K  23[19;9f[K0101
21:06:32 [16;9f[K  23[19;9f[K0101
21:06:33 [16;9f[K  23[19;9f[K0101
21:06:33 [16;9f[K  23[19;9f[K0101
21:06:33 [16;9f[K  23[19;9f[K0101 ... [etc.]
```

("Busy channel on" marks the first of a sequence of "--" flags from the modem that indicate initial detection of a serial-tone signal. These indications can be used by a system controller to implement a collision-avoidance scheme.) Further off-line processing of the raw data file produces a tab-delimited file of SNRs that can be plotted or analyzed further:

```
Time          j       SNR[j] (dB)
21:06:26      1       09
21:06:27      2       09
21:06:27      3       09
21:06:28      4       09
21:06:28      5       10
21:06:28      6       10
21:06:29      7       10
21:06:29      8       10
21:06:30      9       11
21:06:30      10      11
21:06:31      11      11
21:06:31      12      11
21:06:31      13      11
21:06:32      14      23
21:06:32      15      23
21:06:33      16      23
21:06:33      17      23
21:06:33      18      23 ... [etc.]
```

Third-party assessment of serial-tone performance is feasible if the monitored data are not encrypted or otherwise protected by special codes from decoding by listening stations.

3.10. Host-Mode Monitoring of TOR Modes

The host mode in TNCs like the Kantronics KAM and AEA PK-232 facilitates the exchange of commands, responses and data between a PC controller and a TNC in a way that uses simplified (and therefore user-unfriendly) syntax. This allows more rapid (and from a programming standpoint, simpler) control and data exchange. The main application of the host mode so far is in "host" applications that provide easy handling of multiple connected streams, key assignment of frequently exercised commands, help screens, and so on. However, in HF communications using the KAM or PK-232, the host mode offers additional advantages for performance assessment. This requires writing monitoring software that takes advantage of host-mode commands, and doing so should be encouraged.

Host Mode programs often provide the user with data on communications performance (with a connected station or as a third-party listener) that are easier to interpret than flashing LEDs, etc. They do this by sending the TNC special hostmode "query" commands whose responses give nearly-real-time data on PacTOR (or in the case of the

KAM, presumably also GTOR) data rate, Huffman coding, TOR receiving or transmitting state, number of retries and outstanding frames, etc. The host mode application converts the user-unfriendly responses to friendly screen displays of the corresponding status messages.

All frames exchanged between the controlling computer and TNC are delimited by fixed characters that tell the receiving end of the exchange that host-mode commands or data are arriving. In the case of the PK-232, host-mode exchanges are delimited by the Start of Header (SOH, hex 01) and End of Transmission Block (ETB, hex 17) characters. For the KAM, hostmode exchanges are delimited by the Frame End (FEND, hex C0) character.

For the PK-232, host-mode queries from the PC to the TNC have the form

SOH A B ETB,

where A and B stand for a two-letter mnemonic of the corresponding "verbose" command[13]. The PK-232's response has the form

SOH hex4F a b (value) ETB,

where a and b form the query mnemonic and (value) gives the data of the response (Y, N or numerical data).

For the KAM, host-mode queries from the PC to the TNC have the form

FEND?FEND,

where FEND is hex C0. The KAM's response frames have the form

FEND?0MSXYFEND,

where ?0 indicates a response to a query, M gives the mode (Packet, FEC, PacTOR, GTOR, etc.), S the "Submode" (connected, disconnected, standby, etc.), X the "Status" (idle, failed CRC, received request for repeat, Huffman compression, data rate, etc.) and Y shows if PTT is active or that a changeover of transmitting and receiving stations is in progress.

These host-mode queries can be invoked by monitoring programs that can inquire after communications status and performance in connected and some monitoring modes. The queries can be sent on schedule or in response to changes in performance indicated by query responses, throughput, numbers of errors, etc. The responses can be logged for further analysis via screen capture or by having the monitoring program itself write them to a file.

4. Network Assessment

I've defined a network as a set of at least three HF stations, which often means that stations can use automatic relaying. Although auto-assessment of performance by

---

[13]The PK-232 can also send queries about the current operating mode (Packet, AMTOR, FEC, etc.) and link status (AX.25 v1 or v2, number of unacknowledged packets, retries, etc.). A Data Polling command allows one to query the PK-232 on a regular basis as to its status, whether it has changed or not.

stations involved in true HF network communications, rather than "potential" networking (see the discussion of the IPS in Section 5), is possible, I don't know of any networks that are currently doing it on a regular basis[14]. Most of this section will therefore cover third-party assessment.

4.1. AX.25 Networks (Monax25, PACKHACK, PacketTracker, MacAPRS)

Since AX.25 packet radio is one of the oldest data transmission modes used in HF networking, most of the software for third-party assessment of networks pertains to monitoring of AX.25 traffic.

4.1.1. Monax25

Monax25 (for "Monitor AX.25") is a set of C-programs written in the late 80s by Skip Hansen (WB6YMH) and Harold Price (NK6K), which allow real time monitoring and recording of AX.25 packet traffic on one frequency. The programs were written to allow "global" assessment of VHF or HF LAN traffic patterns, where the "local" in LAN means "within receiving distance."

All the Monax programs use the DOS command-line interface. The real-time monitoring is performed by a program called STATS.EXE which has fast, assembler-language modules that handle serial-port I/O with a TNC operating in the KISS mode. STATS displays packet, NET/ROM and TCP/IP address, data and control fields, including the retry flag, in real time on the standard output (screen), and dumps summary data to a time-tagged log file every five minutes (this interval can be easily modified by changing and recompiling the available source code). STATS can also record data-carrier-detection (DCD) activity (monitoring true DCD activity requires a slight modification of older TNCs). Here's a snippet from the beginning of such a log file, recorded in February, 1995, on 14.105 MHz:

```
T,791943670
F,0,0,0,0,0,0,0,0,0,0,4268
T,791943972
F,4,75,3,58,4,0,0,0,0,0,5172
C,VE1CRS-7,N9BMS,0,240,1,1,1,4,4,6,6,6,75,75,791944053,0,0,0,0,0,3,0,1,0,0,0,2,1,0,0,0,0
T,791944274
F,27,503,16,310,26,1,0,0,0,0,5175
C,W9SL,WG9V-7,0,0,0,0,0,3,3,0,0,0,51,51,791944285,0,0,1,0,0,2,0,0,0,0,3,0,0,0,0,0,0
C,KI5DQ-7,KA9ALO,0,240,1,3,3,14,14,2,6,6,247,247,791944322,1,0,0,0,0,10,0,3,0,0,9,2,1,0,0,0,0
C,BEACON,N9BMS,0,240,1,1,1,1,1,34,34,34,52,52,791944350,0,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0
C,KA9ALO,KI5DQ-7,0,0,0,0,0,1,1,0,0,0,17,17,791944367,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0
C,W4KBS,KC4FS,0,0,0,0,0,6,6,0,0,0,102,102,791944492,0,0,1,0,0,5,0,0,0,0,4,1,0,0,0,0,0
C,VE1CRS-7,N9BMS,0,0,0,0,0,2,2,0,0,0,34,34,791944510,0,0,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0
```

The labels at the beginnings of the log-file lines stand for T(ime), F(requency) and C(ircuit). A circuit is an AX.25 Level-2 connection between a pair of stations. Time records appear about every five minutes. Frequency records follow the time records and contain total packets, bytes, unique bytes, and several other statistics on traffic on the monitored frequency during the previous five minutes. Circuit records contain, for each connection monitored during the previous five minutes, the to- and from-calls, total data packets, time the circuit was monitored, the numbers of packets of various sizes, etc. (For a detailed rundown of the recording format see the documentation that accompanies

[14] I'd be happy to hear of examples, perhaps from operators of PacTOR, CLOVER or GTOR BBS message-forwarding stations.

the program archive, which can be found on various bulletin boards and Internet FTP sites.)

The log file can be analyzed off-line with other programs. The next one that's normally used is called REPORT, which combines and summarizes log information to produce one record per five-minute recording interval. The most common REPORT option is called CIRCUIT, which produces for each interval a record containing the interval's time stamp, the number of unique to-from circuits, the number of "user" circuits (excluding beacons, etc.), number of packets on the channel, etc. The CIRCUIT output corresponding to the above log excerpt is

```
791943670,0,0,0,0,0,0,0,0,0,0,0
791943972,1,1,4,0,0,2,0,0,75,6,0
791944274,6,5,27,2,18,4,0,0,503,36,0
```

The RR option of REPORT produces shorter, CIRCUIT-like records consisting of total channel packets, non-digipeated info packets and non-digipeated <<RRn>> frames in the log file:

```
791943670,0,0,0
791943972,4,1,3
791944274,27,3,20
```

Another CIRCUIT option called RAW produces a detailed, circuit-by-circuit summary of each connection during the interval, including a sorting of packets by length, which can be used to make histograms. The RAW record for the second interval above (which contains only one circuit) is:

```
Time Stamp Sat Feb 04 20:26:12 1995

F Total    Total    Unique   Unique   %DCD   %<32  <64   <128  <256  >256
  Packets  Bytes    Packets  Bytes     ON
  4        75       3        58       0.0    100.0 0.0   0.0   0.0   0.0
N9BMS      > VE1CRS-7    ------- I frames --------  ------ All frames -------
             5           Total    NotDigi  Unique   Total    NotDigi
                         Packets  Packets  Packets  Packets  Packets
                         Bytes    Bytes    Bytes    Bytes    Bytes

                         1        1        1        4        4
                         6        6        6        75       75

                         sabm     ua       disc     dm       rej      rr
                         0        0        0        0        0        3

                         rnr      i        ui       frmr     poll     final
                         0        1        0        0        0        2

                         ndigi    <32      <64      <128     <256     >256
                         0        1        0        0        0        0
```

Further programs in the Monax25 suite perform averaging of REPORT records and totaling of log records by station heard and digipeater heard. As can be seen, the comma-delimited output of the REPORT program, in particular, can readily be processed by a spreadsheet or other software to further analyze the logged channel traffic.

4.1.2. PACKHACK

PACKHACK, written in Pascal by Bill Bradford (K7EA), is a much simpler, but still useful, version of Monax25. It analyzes a captured packet stream sent to a terminal with recording capabilities by a TNC set with MON, MCOM, MCON, MALL and MRPT all ON. When you run PACKHACK offline, it prompts you for the name of the captured data file. After filtering out offending ctrl-Z's, PACKHACK sends a table to the screen that lists, by station callsign, the total number of packets received, and the numbers of I, RR, UA, D and REJ packets. The table is not written to a file, but can be saved using a screen-capture utility. Here are the results of a recent PACKHACK analysis of a few minutes' worth of data on a 20m packet channel:

```
The PACKHACK Chronicle for file:  pkth.in

 Originating
     Station      Total            Packet Frame Type
    Callsign    Packets      I     RR     UA     D     REJ

      VE3AZD         10      6      0      0     2      0
      VP9KG-7        11      9      0      0     2      0
      N4BG-15         1      0      0      1     0      0
      N0MFJ-7         9      1      0      0     8      0
    VE3AZD-15        19      0      0      0     0      0
        N0RSU         6      2      0      4     0      0
        VP9KG         8      0      0      0     4      0
        N0MFJ        15     10      0      1     2      0
        AB4UD        39      3      0      0    14      0
        W4KBS        13      0      0      0     0      0
        N9BMS         1      1      0      0     0      0
         WT6B         5      4      0      0     1      0
         WN3Z         1      1      0      0     0      0
       WB6OTO         4      2      0      0     0      0
```

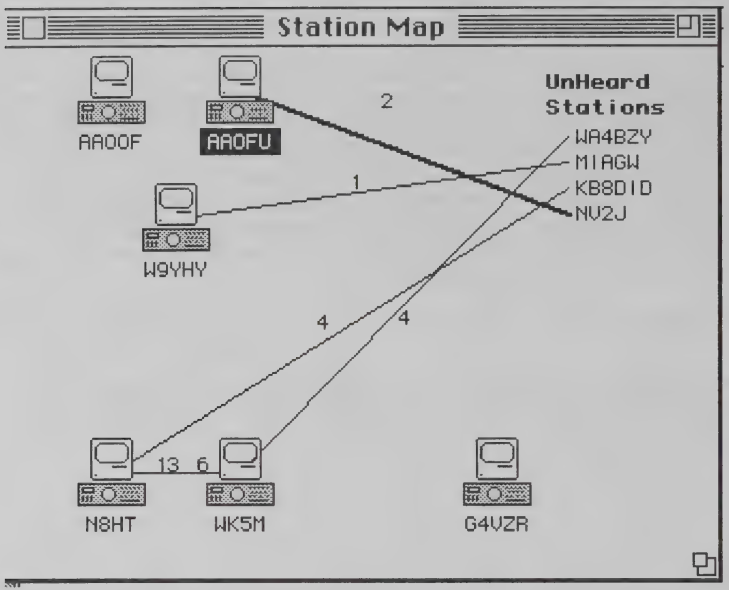
Further details on the program are in the documentation file that comes with the distribution, which can be found at ham radio bulletin boards and Internet FTP sites.

4.1.3. PacketTracker

PacketTracker is an AX.25 (including NET/ROM) monitoring program written by Mark Sproul (KB2ICI) that uses the Macintosh graphical user interface (GUI) to display and record network traffic on one channel in real time. Unlike Monax25, PacketTracker uses the normal TNC interface for access to channel activity detected by the TNC. Details of the program come with the distribution, which can be downloaded over the Internet from ftp.tapr.org and several bulletin boards.

The program displays information in four windows, any combination of which can be displayed at the same time. There is a MAP window, which shows individual connections between pairs of stations, a STATION LIST window, which shows all monitored stations, a STATION INFO window, which gives details of activity for particular stations, and a BAR CHART window, which displays graphs of relative "channel loading," or "channel utilization" as functions of time.

Here's a screen capture of a PacketTracker MAP display after about five minutes of monitoring 14.103 MHz on a recent Sunday afternoon:



When a station sends a packet to another station, a thick line is drawn between them. After 30 seconds the line becomes thin, and it goes away after a user-settable time. Near these lines are written the numbers of sent and received packets. A connection with more than 10% retries shows up as a dashed line[15]. Lines that extend only half way between stations indicate that one of the shown stations has timed out. If no transmission's been heard from a station, it's listed in the UnHeard column. In this example, no lines extend from AA0OF and G4VZR because they have sent beacons and are not connected to anybody. Connection maps can be saved and stations on them can be permanently attached or moved to give the display geographical significance.

The STATION LIST window displays the currently active stations, any aliases, the number of sent packets, the percentage of all monitored packets each station's packets make up and the age since last transmission. After the ages are flags that indicate the station that has most recently transmitted and whether or not it's been heard.

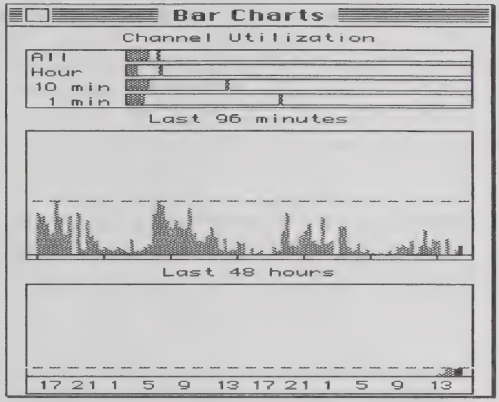
The STATION INFO window gives details for a particular station: its call, alias (if any), number of transmitted packets, number of retries (for some TNCs), age of last transmitted packet, ID string (if any) and the station the selected one is connected to (if any).

One of the most interesting and useful windows shows BAR CHARTS of "channel usage," which is given in terms of the number of characters received from the TNC during the last minute, ten minutes, and hour, and since the program was started, along with the maximum number registered. Other graphs show the usage as a function of time during each of the last 96 minutes and during each of the last 48 hours, along with the maximum recorded usage. Full scale in each case represents ideal usage, presumably a character sent in every character-length time slot. Because of the somewhat arbitrary

[15]This does not apparently happen with KAMs because of the way they format retried frames sent to the computer. KB2ICI has said he'll fix this in a future release.

scales on these graphs, they are best used to assess relative usage. Future versions of the program should perhaps make usage more precise and label axes accordingly.

Here are the bar charts for monitored traffic on 10.14887 MHz LSB on April, 23, 1995, from 13-16 GMT[16]:



PacketTracker will also write monitored data to tab-delimited files that can be analyzed with spreadsheets or other software. These files give cumulative statistics by station, and hourly statistics on traffic from all stations.

4.2. APRS Monitoring with MacAPRS

Although it is not normally used for performance assessment, the Macintosh version of Bob Bruninga's (WB4APR) Automatic Packet Reporting System (APRS), written by Keith and Mark Sproul (WU2Z and KB2ICI), produces a channel usage bargraph similar to PacketTracker's, and a "When Heard" table that lists the number of monitored APRS packets from each station during each hour of the day. The bargraph data and When Heard table can be saved to files for further analysis. For details see the documentation that accompanies the MacAPRS distribution (at ftp.tapr.org).

4.3. NET/ROM Networks

NET/ROM network traffic can be monitored by the same means as point-to-point traffic; namely, with the AX.25 MONITOR command and its variants, and with the TCP/IP TRACE command and its variants. In addition to these, one can query NET/ROM *nodes* (after connecting to them) as to their (local) down- and uplinks by sending them the USERS command. NET/ROM nodes reachable from a particular node (and not "locked"[17]) can be ascertained by sending the node in question the NODES command. Finally, certain (generally fixed) communications parameters used by a particular node can be ascertained by sending the node a PARMS or STATS command, depending on the NET/ROM software package (BPQ, TheNET, etc.) used by the node.

---

[16]Although recorded near the 30m APRS frequency, this is not APRS traffic; see Section 4.2.

[17]Locking nodes and routes defeats NET/ROM's automatic route reporting functions.

4.4. PacTOR/GTOR/CLOVER Networks

PacTOR network traffic can be monitored and recorded in the usual way at a PacTOR-equipped station in the PacTOR LISTEN mode. Link turn-arounds are usually labeled with station callsigns, which facilitates keeping links separate. Host-mode commands can be used to record data rates, turn-arounds and Huffman compression.

GTOR traffic in the (AMTOR) FEC mode can be similarly monitored, but the signal-processing required to monitor connected stations requires software dedicated to decoding the Golay codewords used for error correction in the GTOR ARQ mode[18]. Host-mode commands can presumably[19] be used to simplify recording of data rates, turn-arounds, etc.

CLOVER's built-in monitoring facilities allow detailed monitoring and recording of performance of stations operating in the FEC mode. The close coordination of stations adapting themselves to channel conditions in the ARQ mode makes monitoring of networks operating in that mode difficult if not impossible[20].

Since the ARQ modes of PacTOR, GTOR and CLOVER are the ones in which truly adaptive HF communications occur (i.e., adaptation to channel conditions using *feedback*), network monitoring of ARQ traffic is a project worthy of attention from software writers. Note, however, that such monitoring is complicated by the fact that in most well designed adaptive HF networks, more than one frequency will be used.

5.1. More Advanced Monitoring

Most recent work on adaptive HF networking (mainly by companies interested in military or other government sales) has been carried out under the aegis of the HF Radio Subcommittee of the National Telecommunications and Information Administration (NTIA), a part of the US Department of Commerce. The subcommittee is one of two government organizations[21] that are concerned with the development of standards for advanced HF digital communications. These standards specify various aspects of packet radio protocols designed specifically to get maximum error-free throughput from HF channels. The work ranges from the study of improvements in protocols for link establishment (including methods for over-the-air sharing of link quality measurements) to encryption of ALE signals and deciding the best way to adjust frame sizes and frame window sizes for maximum throughput in the ARQ mode when using an advanced serial-tone modem.

A typical station running such a protocol might have, in addition to transceiver and antenna, a fast PC to run the whole system, an ALE modem for linking, a device for encrypting data and a data modem with error-correction and an equalizer for transfer of encrypted data over good channels found by the ALE modem. The adaptive packet protocol is run on the PC. (Stations that are part of a new HF network used in the US Air

---

[18]The v8.0 GTOR upgrade apparently allows direct monitoring.

[19]The GTOR mode is new, and GTOR traffic that can be monitored is relatively rare. Readers who have monitored more than one GTOR ARQ link at the same time in the host mode or otherwise are invited to send me the details.

[20]News of successful monitoring of CLOVER ARQ traffic on more than one link is welcome.

[21]The other is run by the Army at Ft. Huachuca, Ariz.

Force's Information Processing System (IPS) have this setup. IPS stations can participate in true adaptive networking, in which stations exchange channel quality and network status information, and adjust data rates, frame lengths and message routes on the fly for highest throughput.)

Monitoring the performance of such a station during its development requires access to the asynchronous control lines between the PC and the ALE modem (two channels), the asynchronous control lines between the PC and the data modem (two channels) and the data lines (usually synchronous) between the PC and the crypto device (two channels). Sometimes request-to-send and clear-to-send signals (called "discretes") to or from the crypto device are also monitored. The monitoring thus involves at least six channels.

All this monitoring can be done from a fast PC with a big hard drive. The monitoring PC is very busy and is separate from the PC controlling the station. The monitoring PC uses serial- and parallel-I/O cards[22] mounted in its expansion slots to collect data from each channel. Since the data arrive unpredictably on the various channels, an arrangement for rapid sampling of each channel may be needed so as not to miss any data. One method is to use a ring buffer and "interrupt service routines" to poll the channels and react to any data that have arrived on them. Although this must be done rapidly, it is in practice feasible since usually only one of the modems or devices sends data at a time.

Data from each channel are usually written to their own labeled and dated file. It's almost always a good idea to have the monitoring program add *time tags* to the captured data files on a line-by-line or event-by-event basis.

5.2. Statistical Analysis of Performance Data

While no fixed rules for writing performance analysis software can be written, here are some general guidelines:

Most performance analysis can be done off-line; that is, by analyzing files of data captured by a terminal program (possibly using a scripting language) or a program written specifically for data capture. In rare cases, a certain amount of on-line (real-time) analysis must be done to process incoming data before they go to a file if they arrive too fast, or there are too many to store, or their format is not suitable for immediate writing to a text file.

Analysis of a captured file often starts with *parsing*, which consists of line-by-line reading of the file and tests for certain characters or strings of characters that indicate events of interest ("linked to W3XYZ," "IRS," "Huffman on," "RETRIES=4," etc.). The lines, or parts of lines, containing these "flags" are then often written to a new file for further analysis or record keeping.

Numerical data in the raw data file, which are often located with a parsing test (e.g., data rates, numbers of retries, channel qualities, interleaver depths, etc.) are also captured during parsing. They can be written to a tab-, space- or comma-delimited file for plotting or further analysis, or analyzed during the parsing itself. (The shareware UNIX *gnuplot* program, which has been ported to PCs and Macs, can simultaneously open a data file and a file of formatting commands created by a parsing program itself; this allows one-step plotting of parsed captured data.)

---

[22]These cards can handle several channels simultaneously and cost two or three hundred dollars a piece.

All modern high level languages like C, C++, FORTRAN and Pascal offer extensive libraries of functions for reading or writing characters and strings, and locating or comparing particular characters and strings. These functions make parsing relatively painless.

Statistical analysis of a sample $x_1, x_2, ..., x_N$ of N random measurements (signal-to-noise ratios, LQA scores, etc.) often starts with calculations of the sample's basic statistics:

its *mean* $m = (1/N)\sum_{i=1}^{N} x_i$, its (unbiased) *standard deviation*

$$s = \sqrt{\sum_{i=1}^{N} x_i^2 / (N-1) - Nm^2 / (N-1)},$$

and the standard deviation (*standard error*) of its mean $s / \sqrt{N}$.[23] The sample standard deviation (and thus also the standard error of the mean) measures the spread of the data about their mean, and therefore how well the mean alone summarizes the data. The standard deviation and sample size are used to form *confidence intervals* for the mean, which is another way to assess how well the mean summarizes the sample.

More sophisticated counting and analysis of bit- or character errors, and numerous other measures of performance, are restricted only by the properties of the system being studied and the analyzer's imagination and programming skills. Fortunately for us hams, there are plenty of talented programmers in our ranks. I hope this article sends a few of them to their compilers, and encourages everybody else to get down and dirty with digital HF.

Acknowledgments.

I am grateful to Bob Levreault (W1IMM) and Dave ("Doc") Willard (W1EO) for comments on the paper that clarified several points.

Appendix: Text File Comparison.

As an example of the comparison of a received text file (generally containing errors or "missed character" symbols of one kind or another) and a stored correct file, I've written a relatively simple C-program that prompts the user for the names of received and correct files and performs such a comparison. The program counts the numbers of characters (including end-of-file characters) in each file, finds the carriage-returns in each file, and counts the number of printable, upper-case and "distinguished" (a user input) characters in the received file. A distinguished character might be a missed character symbol, as with AMTOR. The program also lists all character positions in the received file for which there is a different character at the corresponding position of the correct file. A user-prompt also allows a string-by-string comparison of the files, which helps determine the number of missmatched words; this is useful when the files don't have the same size. Here's the output of the program for files called test4 (correct) and test4r (a made-up received file). The correct file:

[23]The N-1 in place of the expected N in these estimates offers the minor advantage that it makes the estimates *unbiased* (they approach the true population statistics as N -> ∞).

Cicero considered Syracuse the "most beautiful of Greek cities," although by his time it hadn't been ruled by Greeks for four centuries.

The received version is

Cicero con-idered Syracuxe the "most b456tiful of Greek cities," although by his time it haxn't been ruled by Gr--ks for four centuries.

The distinguished character was chosen to be "x" and the string-by-string option was not exercised.

The program output is

```
Correct input file is [test4].
Received input file is [test4r].

Correct/received file has 141/141 characters.

Locations of <CR>s:
Correct file has <CR> at position 74.
Correct file has <CR> at position 138.

Received file has <CR> at position 74.
Received file has <CR> at position 138.

Printable & upper-case character count:
Received file has 114 printable and 4 upper-case character(s).

Distinguished character count:
[Distinguished character is "x"]
Received file has 2 distinguished character(s).

Whole-file analysis:
Rcvd char. 10 is "-"; Correct char. 10 is "s"
Rcvd char. 24 is "x"; Correct char. 24 is "s"
Rcvd char. 38 is "4"; Correct char. 38 is "e"
Rcvd char. 39 is "5"; Correct char. 39 is "a"
Rcvd char. 40 is "6"; Correct char. 40 is "u"
Rcvd char. 93 is "x"; Correct char. 93 is "d"
Rcvd char. 114 is "-"; Correct char. 114 is "e"
Rcvd char. 115 is "-"; Correct char. 115 is "e"

Received file has 8 missmatch(es); 5.7% of characters.

Line-by-line analysis:
Rcvd line 1 has <CR> at posn 74 [Corr. char. is 0xd], 74 chars and 5 missm(es)
Rcvd line 2 has <CR> at posn 138 [Corr. char. is 0xd], 63 chars and 3 missm(es)
```

Such comparison files can be submitted to further analysis if a large number need to be summarized.

Proposed Recommendation for: Hierarchical Addressing Protocol

Authors: Dave Wolf, WO5H
Greg Jones, WD5IVD
Roy Engehausen, AA4RE
Hank Oredson, W0RLI

Date: August 30th, 1994 (Revised December 18, 1994)

Comment Period on Table Changes: September 1st, 1994 through January 31st, 1995

Send Comments and Supplemental Table Input to:
Dave Wolf, WO5H, TAPR BBS SIG Chair
Packet: wo5h@wo5h.#dfw.tx.usa.noam
Internet: dwolf@tcet.unt.edu
Fax: (817) 295-6232

Introduction

The TAPR BBS Special Interest Group recommends the adoption of the x.3.4 hierarchical address protocol.

After discussion of previous articles on hierarchical addressing standards (1,2) and taking into account international issues of regional/state name sizes, the TAPR BBS Special Interest Group recommends the adoption of the x.3.4 standard on an international basis. x is defined as 2, 3, or 4 letter region names as defined by the country.

Examples of x.3.4:
@WA6GVD.CA.USA.NOAM
@EA2CMO.EAZ.ESP.EURO
@F5JGK.FAQI.FRA.EURO

Regional identifiers may be duplicated in different countries (i.e. AK, Alaska, USA, could be used in another country as a regional identifier); however, Country and Continental identifiers SHOULD NOT be used as regional names.

It is important to note that there is a distinct and significant difference between HIERARCHICAL ADDRESSES and FLOOD DESIGNATORS. Hierarchical address elements are common to all messages (bulletins, P and T types) and are the foundation of the digital forwarding system. Flood designators are used for routing and filtering bulletins. Geographical flood designators are likely based upon hierarchical address elements. It is therefore important that any attempt to establish standards concentrate first on hierarchical address elements. Standards for flood designators can follow.

It is the purpose of this document to establish a basic framework upon which to base a logical hierachical addressing structure. Attached Supplemental Tables will be changed as necessary to reflect current configurations within the international BBS network.

Hierarchical Routing Syntax Summary

This summary uses a modified Backus-Naur form to summarize the syntax for hierarchical addressing.
[] = optional

@hierarchial_address
bbs.[#octothorpe.][region.]country.continent

bbs
valid callsign as defined by local communications authority

#octothorpe.
#area.[#octothorpe.]

#area
local area identifier as defined by the sysops of the local region

See Supplemental Table 4 for list of current area identifiers

region
2,3, or 4 character region identifier (for state, province, department, prefecture, etc.)
as defined by sysops within each country.

See Supplemental Table 3 for list of region identifiers

country
3 character country identifier as defined by ANSI X.12 and EDIFACT.
Published in ISO 3166-1981(E/F).

See Table 2 for country identifiers

continent
4 character continental identifier.

See Table 1 for continental identifiers.

Examples:
F6CNB.#SETX.TX.USA.NOAM

KB7WE.#WWA.WA.USA.NOAM
OH6RBV.#VAA.FIN.EURO
SK2AT.AC.SWE.EURO
OH6RBG.FIN.EURO
KE7KD.#NONEV.NV.USA.NOAM
WX3K.#NOCAL.CA.USA.NOAM

References:

1. Jenkins, Lew (N6VV), Dave Toth (VE3GYQ), and Hank Oredson (W0RLI). International Routing Designators. Proceedings of the ARRL 7th Computer Networking Conference. Columbia Maryland. October 1, 1988. pp. 91-93.

2. Clark, Tom (W3IWI). Some comments on the `H`ierarchical Continent Address Designator. Proceedings of the ARRL 9th Computer Networking Conference. London, Ontario Canada. September 22, 1990. pp. 278-279.

TABLE 1: Continent Identifiers

EURO -- Europe
MEDR -- Mediterranean
INDI -- Indian Ocean including the Indian subcontinent
MDLE -- Middle East
SEAS -- South-East Asia
ASIA -- The Orient

NOAM -- North America (Canada, USA, Mexico)
CEAM -- Central America
CARB -- Caribbean
SOAM -- South America

AUNZ -- Australia/New ZeAC -- Eastern Pacific
NPAC -- Northern Pacific
SPAC -- Southern Pacific
WPAC -- Western Pacific

NAFR -- Northern Africa
CAFR -- Central Africa
SAFR -- Southern Africa

ANTR -- Antarctica

TABLE 2: Country Identifiers

ARG -- Argentina
AUS -- Australia
AUT -- Austria
BEL -- Belgium
BMU -- Bermuda
BOL -- Bolivia
BRA -- Brazil
BRN -- Brunei
BGR -- Bulgaria
CAN -- Canada
CHL -- Chile
CHN -- China
COL -- Colombia
CRI -- Costa Rica
CUB -- Cuba
DNK -- Denmark
DOM -- Dominican Republic
ECU -- Ecuador
EGY -- Egypt
SLV -- El Salvador
FIN -- Finland
FRA -- France
PYF -- French Polynesia
DEU -- Germany
GRC -- Greece
GRL -- Greenland
GTM -- Guatemala
HTI -- Haiti
HND -- Honduras
HKG -- Hong Kong
HUN -- Hungary
ISL -- Iceland
IND -- India
IDN -- Indonesia
IRL -- Ireland
ISR -- Israel
ITA -- Italy
JPN -- Japan
PRK -- Korea, North
KOR -- Korea, South
LBN -- Lebanon
LIE -- Liechtenstein
LUX -- Luxembourg
MYS -- Malaysia
MEX -- Mexico
MCO -- Monaco
MAR -- Morocco
NLD -- Netherlands
NZL -- New Zealand
NIC -- Nicaragua
NOR -- Norway
PAK -- Pakistan
PAN -- Panama
PRY -- Paraguay
PER -- Peru
PHL -- Phillipines
POL -- Poland
PRT -- Portugal
ROM -- Romania
SAU -- Saudi Arabia
SGP -- Singapore
ZAF -- South Africa
ESP -- Spain
SWE -- Sweden
CHE -- Switzerland
SYR -- Syria
TWN -- Taiwan
THA -- Thailand
TUR -- Turkey
GBR -- United Kingdom
USA -- United States
URY -- Uruguay
SUN -- USSR ??? (Document needs latest defintions)
VEN -- Venezuela
YUG -- Yugoslavia

END of formal document.

SUPPLEMENTAL TABLES follow, which may be modified on a continuing basis to reflect the changing nature of the amateur digital communications network. It is the express desire of the authors that these tables be 'fleshed out' in time as acceptance of the standard proposed in this document grows. The supplemental tables may be maintained by TAPR and desseminated, as necessary.

Supplemental Tables for Hierarchical Addressing Protocol Document. These tables are to be maintained by TAPR, Tucson Amateur Packet Radio Corporation, based on continuing input from the amateur community.

TABLE 3 (SUPPLEMENTAL): Region Identifiers organized by Country Codes.

ARG -- Argentina
BA -- ??
CF -- ??
AUS -- Australia
??
AUT -- Austria
??
BEL -- Belgium
HT -- ??
LG -- ??
OVN -- ??
WVL -- ??
BMU -- Bermuda
??
BOL -- Bolivia
??
BRA -- Brazil
RS -- ??
SP -- ??
BRN -- Brunei
??
BGR -- Bulgaria
??
CAN -- Canada

AB -- Alberta
BC -- British Columbia
LB -- Labrador
MB -- Manitoba
NB -- New Brunswick
NF -- Newfoundland
NS -- Nova Scotia
NT -- Northwest Territories
ON -- Ontario
PE -- Prince Edward Island
PQ -- Quebec
SK -- Saskatchewan
UT -- Yukon Territory

CHL -- Chile
??
CHN -- China
??
COL -- Colombia
??
CRI -- Costa Rica
??
CUB -- Cuba
??
DNK -- Denmark
??
DOM -- Dominican Republic
??
ECU -- Ecuador
??
EGY -- Egypt
??
SLV -- El Salvador
??
FIN -- Finland
??
FRA -- France
FCEN -- ??
FRPA -- ??
FCAL -- ??
FPDL -- ??
FMLR -- ??
FNOR -- ??
FCOR -- ??
FPOC -- ??
FAQI -- ??
PYF -- French Polynesia
??
DEU -- Germany
BY -- ??
GRC -- Greece
??
GRL -- Greenland
??
GTM -- Guatemala
none
HTI -- Haiti
??
HND -- Honduras
??
HKG -- Hong Kong
??
HUN -- Hungary
??
ISL -- Iceland
??
IND -- India
??
IDN -- Indonesia
??
IRL -- Ireland
??
ISR -- Israel
??
ITA -- Italy
IEMR -- ??
IFVG -- ??
ILOM -- ??
IPIE -- ??
IPUG -- ??
ISAR -- ??
ISIC -- ??
ITAA -- ??
IVEN -- ??
MO -- ??
JPN -- Japan
??
PRK -- Korea, North
??
KOR -- Korea, South
??
LBN -- Lebanon
??
LIE -- Liechtenstein
??
LUX -- Luxembourg
??
MYS -- Malaysia
??
MEX -- Mexico
??

MCO -- Monaco
??
MAR -- Morocco
??
NLD -- Netherlands
??
NZL -- New Zealand
??
NIC -- Nicaragua
??
NOR -- Norway
??
PAK -- Pakistan
??
PAN -- Panama
??
PRY -- Paraguay
??
PER -- Peru
??
PHL -- Phillipines
??
POL -- Poland
??
PRT -- Portugal
CTPT -- ??
ROM -- Romania
??
SAU -- Saudi Arabia
??
SGP -- Singapore
??
ZAF -- South Africa
??
ESP -- Spain

EAA -- Alicante
EAAB -- Albacete
EAAL -- Almeria
EAAV -- Avila
EAO -- Asturias
EABI -- Vizcaya
EAB -- Barcelona
EABA -- Badajoz
EABU -- Burgos
EAC -- La Coruna
EACA -- Cadiz
EACC -- Caceres
EACE -- Ceuta
EACO -- Cordoba
EAH -- Huesca
EASE -- Sevilla
EAHU -- Heulva
EAZ -- Zaragoza
EACR -- Ciudad Real
EACS -- Castellon
EACU -- Cuenca
EAGC -- Gran Canaria
EAGI -- Girona
EAGR -- Granada
EAGU -- Guadalajara
EAHU -- Huelva
EAJ -- Jaen
EAL -- Lleida
EALE -- Leon
EALO -- La Rioja
EALU -- Lugo
EAM -- Madrid
EAMA -- Malaga
EAML -- Melilla
EAMU -- Murcia
EANA -- Navarra (Pamplona)
EAO -- Oviedo
EAOR -- Orense
EAP -- Palencia
EAPM -- Baleares (Palma de Mallorca)
EAPO -- Pontevedra
EAS -- Cantabria
EASA -- Salamanca
EASG -- Segovia
EASO -- Soria
EASS -- Guipuzcoa
EAT -- Tarragona
EATE -- Turuel
EATO -- Toledo
EATF -- Santa Cruz de Tenerife
EAV -- Valencia
EAVA -- Valladolid
EAVI -- Alava
EAZA -- Zamora

SWE -- Sweden
AC -- ??
CHE -- Switzerland
??
SYR -- Syria
??
TWN -- Taiwan
??
THA -- Thailand
??
TUR -- Turkey
??
GBR -- United Kingdom (need a list of routing numbers by county)
#1 -- ??
#2 -- ??
...-- ??
USA -- United States
AK -- Alaska
AL -- Alabama
AR -- Arkansas
AZ -- Arizona
CA -- California
CO -- Colorado
CT -- Connecticut
DE -- Delaware
FL -- Florida
GA -- Georgia
HI -- Hawaii
IA -- Iowa
ID -- Idaho
IL -- Illinois
IN -- Indiana
KS -- Kansas
KY -- Kentucky
LA -- Louisiana
MA -- Massachusetts
MD -- Maryland
ME -- Maine

MI -- Michigan
MI -- Mississippi
MN -- Minnesota
MO -- Missouri
MT -- Montana
NC -- North Carolina
ND -- North Dakota
NE -- Nebraska
NH -- New Hampshire
NJ -- New Jersey
NM -- New Mexico
NV -- Nevada
NY -- New York
OH -- Ohio
OK -- Oklahoma
OR -- Oregon
PA -- Pennsylvania
RI -- Rhode Island
SC -- South Carolina
SD -- South Dakota
TN -- Tennessee
TX -- Texas
UT -- Utah
VA -- Virginia
VT -- Vermont
WA -- Washington
WI -- Wisconsin
WV -- West Virginia
WY -- Wyoming
DC -- District of Columbia
PR -- Puerto Rico

URY -- Uruguay
MVD -- ??
SUN -- USSR ??? (Document needs latest defintions)
??
VEN -- Venezuela

YUG -- Yugoslavia
SRB -- ??

Table 4: (Local) Area Defintions

All readers are suggested to submit their local area defintions for inclusion in the table. Be sure to include the region and country.
Example: #DFW.TX.USA -- Dallas/Ft Worth Texas Area would be an entry in this table.

USA -- United States

AK -- Alaska
#ADAK#AK
#SAK#AK

AL -- Alabama
#BHM#AL
#CENAL#AL
#FHP#AL
#HSV#AL
#MOBAL#AL
#BHM#AL
#CENAL#AL
#FHP#AL
#HSV#AL
#MOBAL#AL

AZ -- Arizona

*.AZ
We do not implement .# do to our network system allowing live connects from one end of the state to another. We agreed this year to not to make any changes because things are still working so well.

FL -- Florida
#MOB#FL
#MLBFL#FL
#NEFL#FL
#NFL#FL - North Florida Section
#OCFFL#FL
#OKLAN#FL - Okaloosa County LAN
#ORLFL#FL - Orlando, FL
#PASFL#FL
#PFNFL#FL
#PMPFL#FL
#PNSFL#FL - Pensacola, FL
#PSJ#FL
#RCCFL#FL
#SRQ#FL
#SRQFL#FL
#STC#FL
#STU#FL
#SUNFL#FL
#SWFL#FL
#TAV#FL
#TLH#FL - Talahassee, FL
#TPA#FL - Tampa, FL
#VENFL#FL
#VRBFL#FL - Vero Beach, FL

#WPBFL#FL - West Palm Beach, FL

GA -- Georgia

#CVL#GA
#DLT#GA
#ETADS#GA
#MAR#GA
#NGA#GA
#NOGA#GA
#NTSGA#GA
#SAV#GA
#SEGA#GA
#SSI#GA
#STK#GA
#SWGA#GA
#VLD#GA
#WAL#GA
#WCGL#GA

IA -- Iowa

#CIA#IA
#ECIA#IA
#KA0STB#IA
#NWIA#IA
#SEIA#IA
#SWIA#IA
#WCIA#IA

IL -- Illinois

#NEIL.IL.USA NORTHEASTERN IL
#NWIL.IL.USA NORTHWESTERN IL
#NCIL.IL.USA NORTHCENTRAL IL
#SIL.IL.USA SOUTHERN IL
#CIL.IL.USA CENTRAL IL
#ECIL.IL.USA EASTCENTRAL IL
#WCIL.IS.USA WESTCENTRAL IL

IN -- Indiana

#CEIN#IN
#CIN#IN
#N0IN#IN
#NCIN#IN
#NEIN#IN
#NIN#IN
#NOIN#IN
#NWIN#IN
#SIN#IN

KS -- Kansas

#NWKS.KS.USA -- Northwest Kansas (21 counties)
#NCKS.KS.USA -- North Central Kansas (15 counties)
#NEKS.KS.USA -- North East Kansas (21 counties)
#SEKS.KS.USA -- South East Kansas (15 counties)
#SCKS.KS.USA -- South Central Kansas (15 counties)
#SWKS.KS.USA -- Southwest Kansas (18 counties)

KY -- Kentucky

#MIDKY#KY
#NCKY#KY
#NEKY#KY
#NWKY#KT

NC -- North Carolina

#AVL#NC
#BUY#NC
#CLT1#NC
#DUR#NC
#EASTNC#NC
#GAS#NC
#GSO#NC
#HKY#NC
#HKY1#NC
#ILM#NC
#INT#NC
#LBT#NC
#MEB#NC
#RTP#NC
#RTP#NC

ND -- North Dakota

#DL#ND
#GFK#ND
#JAM#ND
#MOT#ND
#PAR#ND
#RL#ND
#SEND#ND

NE -- Nebraska

#ASHBY#NE
#ENE#NE
#NENE#NE
#SCNE#NE
#SENE#NE
#WNE#NE

NH -- New Hampshire

#NNH#NH
#SCNH#NH

NJ -- New Jersey

#NNJ#NJ
#SNJ#NJ

NV -- Nevada

#NENEV#NV Covers Elko and Ely Nevada

#NONEV#NV Covers Reno, Carson, Fallon, Lake Tahoe Nevada

#SONEV#NV Covers Las Vegas BBSs

NY -- New York

#CNY#NY
#ENY#NY
#NENY#NY
#NLI#NY
#NNY#NY
#NYC#NY
#WNY#NY

OH -- Ohio

#CIN#OH
#CMH#OH
#DAY#OH
#EOH#OH
#NCOH#OH
#NEOH#OH
#NOH#OH
#NWOH#OH
#SEOH#OH
#SWOH#OH
#TOL#OH

OK -- Oklahoma

#NCOK#OK
#NEOK#OK
#OKC#OK
#SCOK#OK

PA -- Pennsylvania

#EPA.PA.USA -- East PA

#WPA.PA.USA -- Western PA

#cpa
#nepa
#sepa
#ncpa
#scpa
#nwpa
#swpa

TX -- Texas

#AUS
#CENTX
#DFW
#ETX
#NTX
#SAT
#STX
#SETX
#WTX

WA -- Washington

#SWA#WA
#SEA#WA
#SEWA#WA
#SPOKN#WA
#VANC#WA
#WWA#WA

WI -- Wisconsin

#CFL#WI
#CWI#WI
#DUR#WI
#ECWI#WI
#JNSVL#WI
#MKE#WI
#NOWI#WI
#NWI#WI
#SCWI#WI
#SEWI#WI
#TMH#WI
#WWI#WI

WV -- West Virginia

#NEWV#WV
#NWV#WV

WY -- Wyoming

#NWWY#WY
#RAWL#WY
#SEWY#WY
#SOWY#WY

---- END -----

Thanks to these folks for input and/or comments on this project:

Angel A. Padin de Pazos EA1QF, Jim Brooks K0JJV, EA1DOF, Bill Barnes N3JIX, Dirk G1TLH, Dan VE2PNK, Bill Healy N8KHN, Bob Wilkins, Hank K2UVG, Jim Fortney K6IYK, Barry WA0RJT, Daniel J. Meredith N7MRP, Karl K5DI, Ron VE1AIC and many others.

Published by: THE AMERICAN RADIO RELAY LEAGUE